066331.P058 *Patent*

UNITED STATES PATENT APPLICATION

FOR

METHOD AND APPARATUS FOR FACILITATING DATA REPLICATION USING OBJECT GROUPS

INVENTORS:

HARRY A. SUN
BENNY SOUDER
PETER LIM

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026

(408) 720-8598

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EH166607791 US

Date of Deposit June 21, 1996

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Drew R. Herndon
(Typed or printed name of person mailing paper or fee)

Drew R Herndon
(Signature of person mailing paper or fee)

METHOD AND APPARATUS FOR FACILITATING DATA REPLICATION USING OBJECT GROUPS

FIELD OF THE INVENTION

The invention relates generally to the field of database management systems. More specifically, the invention relates to data replication in a distributed database system.

BACKGROUND OF THE INVENTION

Modern data processing systems are distributed over many interconnected processing nodes (sites). The efficiency of these distributed systems depends not only upon the processing power of each independent node, but also upon the ability of the system to efficiently access the information required by the user. For example, a node can access data that resides on storage within the node (perform a "local access") much faster than it can access data stored in another node (perform a "remote access"). To take advantage of the efficiency of local access operations, some systems allow the same set of data to be maintained on multiple nodes. The mechanism for maintaining multiple copies of the same data on multiple nodes is referred to as data replication. One mechanism for performing data replication is described in U.S. Patent Application No. 08/479,075 entitled "Method and Apparatus for Data Replication" filed on June 6, 1995 by Sandeep Jain et al. Using data replication, multiple replicas of data can exist in more than one database in a distributed system.

A "schema" can be thought of as a group of formatted, stored data. Some database management systems utilize several schemas. For example, the physical data in a database can be partitioned into several users, each user represented with a schema. In this example, a schema acts as a physical division mechanism defining boundaries between groups of physical data. Other database management systems recognize only one schema. In this type of system the physical data in the database all resides in the same schema. In either case a schema represents a physical structure of the database.

The lowest level at which replication can be performed is referred to as the unit of replication. One approach to performing data replication employs "schemas" as the unit of replication. This approach will be discussed with respect to a database management system that recognizes several schemas. By using schemas as the unit of replication, in certain circumstances this approach can replicate no less than all of the replicated objects that are members of a given schema. For example, assuming each object in a schema is required to exist at more than one node, if another node needs local copies of a subset of the objects, all the registered replicated objects in the schema must be copied to that node, even if the node does not require all the objects.

Another disadvantage of using this prior schema-level data replication is that all schemas at every site have to have the same replicated content. In other words, every replica of a given schema must contain all the member objects registered as replicated objects regardless of whether or not each node requires the replicated data modifications of all the replicated objects. These disadvantages result from tying replication to schemas as will be discussed further with respect to **Figure 1**. These

disadvantages arise due to the underlying assumption that all schema replicas require the same content at each site. When this underlying assumption does not hold true, a system that controls replication at the schema-level will be faced with one of two limitations: (1) generation of more replication traffic than necessary, or (2) schema proliferation. Replication traffic is the amount of data that must be passed between the nodes in the distributed system to maintain the replicated data. Increased network traffic reduces the available bandwidth to perform other activities in the distributed system.

Figure 1 illustrates a distributed processing system employing a schema-level data replication method. As illustrated by the following example, with this approach, excess replication traffic is generated when different nodes have different replication needs. Referring to **Figure 1**, a distributed data processing system comprises node 1, node 2, and node 3 coupled together with network links 120, 125, and 130. A first schema 110 containing objects A, B and C is resident on node 1. Also resident on node 1 is a second schema 115 containing objects D, E and F. In this example, node 2 requires access to objects A, C and D, but does not require access to objects B, E or F. Node 3 requires access to objects B, C, E and F but does not require access to objects A or D. Dashed lines represent an operation that requires the object to be read or modified. Nodes containing replicated data are said to be participating in replication.

For node 2 to have local access to objects A and C, the data specified in the first schema 110 must be replicated on node 2. Likewise, for node 3 to have access to objects B and C, the data specified in the first schema 110 must be replicated on node

3. Upon replicating the first schema 110 to node 2 and node 3, node 2 will contain a replica 140 of the first schema 110 and node 3 will contain a replica 150 of the first schema 110. Similarly, the data specified in the second schema 115 will have to be replicated on node 2 to provide for local access to object D. Further, the data specified in the second schema 115 will have to be replicated on node 3 to provide for local access to objects E and F. Upon replicating the second schema 115 to node 2 and node 3, node 2 will contain a replica 145 of the second schema 115 and node 3 will contain a replica 155 of the second schema 115. The inefficiencies of this method become apparent when one considers node 2 will unnecessarily receive and store replicated modifications for objects B, E and F and node 3 will receive and store modifications regarding objects A and D for which it has no need. This method also has security consequences. For example, the mere presence of object B at node 2 may pose a security risk.

One solution to replication of unnecessary data modification would be to define schemas at a lower level of granularity. For example, separate schemas could be defined for objects A, B, and C (e.g., one schema would comprise only object A, another schema would comprise only object B, and another schema would comprise only object C). Breaking schemas down in this manner would solve the problem of unnecessary replication. Each node could copy only the objects to which it required access and no more. However, this approach leads to a dramatic increase in the number of schemas relative to the number of objects ("schema proliferation"). The increased number of schemas increases the complexity and burden of administering schemas for the database administrator (DBA). Schemas are typically composed of

groups of data that are logically related in a way that has little to do with replication. For example, schemas can be used to impose security barriers, allowing different security levels to be assigned to individual schemas. Breaking schemas up for the purpose of replication would destroy the groupings for which the schemas were initially established.

It is, therefore, desirable to provide an apparatus and method for facilitating data replication that is independent of schemas and their related constraints. It is also desirable to allow a user to assign logical names and administer replication at an arbitrary group level that will allow users to distribute objects at a higher level of granularity than feasible with the prior schema-level data replication method.

SUMMARY OF THE INVENTION

A method and apparatus for replicating data in a computer system having a plurality of sites is disclosed. One or more sets of formatted data is stored at a first site. A mapping is created between subsets of data within the sets of formatted data and one or more object groups. One or more remote sites are specified for each of the object groups.

For each of the object groups, a replica of each subset of data that is mapped to the object group is created at the one or more specified remote sites.

When a data modification is detected to a subset of data that has been mapped to one of the one or more object groups, the data modification is propagated to the objects group's one or more remote sites.

According to one aspect of the invention, the one or more sets of formatted data stored at the first site includes a first and a second set of formatted data. The step of creating the mapping then includes creating a mapping of a first subset of data residing in the first set of formatted data to an object group and creating a mapping of a second subset of data residing in the second set of formatted data to the same object group.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is an illustration of a distributed system performing data replication using schemas as the unit of replication.

Figure 2 is an example of a typical architecture of a node within a distributed processing system upon which one embodiment of the present invention can be implemented.

Figure 3 is a database data dictionary design for supporting a method of specifying a replication environment and replicating data according to one embodiment of the present invention.

Figure 4a is a flow diagram illustrating a method of specifying a replication environment and replicating data according to one embodiment of the present invention.

Figure 4b is a portion of the flow diagram of **Figure 4a**.

Figure 5 is an illustration of a distributed data processing system which supports data replication according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for facilitating data replication is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

HARDWARE OVERVIEW

Referring to **Figure 2**, a computer system is shown as 200. The computer system 200 represents a node (site) within a distributed processing system upon which the preferred embodiment of the present invention can be implemented. The hardware architecture of nodes within the distributed processing system can be varied and diverse. There is no requirement in the present invention that each node have equivalent and compatible processing systems. It is only necessary that each node of the distributed processing system be able to communicate on a network or some communication path coupling the nodes together. Computer system 200 comprises a bus or other communication means 201 for communicating information, and a processing means 202 coupled with bus 201 for processing information. Computer system 200 further comprises a random access memory (RAM) or other dynamic storage device 204 (referred to as main memory), coupled to bus 201 for storing information and instructions to be executed by processor 202. Main memory 204 also may be used for

storing temporary variables or other intermediate information during execution of instructions by processor 202. Computer system 200 also comprises a read only memory (ROM) and/or other static storage device 206 coupled to bus 201 for storing static information and instructions for processor 202. Data storage device 207 is coupled to bus 201 for storing information and instructions.

A data storage device 207 such as a magnetic disk or optical disc and its corresponding drive can be coupled to computer system 200. Computer system 200 can also be coupled via bus 201 to a display device 221, such as a cathode ray tube (CRT), for displaying information to a computer user. An alphanumeric input device 222, including alphanumeric and other keys, is typically coupled to bus 201 for communicating information and command selections to processor 202. Another type of user input device is cursor control 223, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 202 and for controlling cursor movement on display 221. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), which allows the device to specify positions in a plane.

Alternatively, other input devices such as a stylus or pen can be used to interact with the display. A displayed object on a computer screen can be selected by using a stylus or pen to touch the displayed object. The computer detects the selection by implementing a touch sensitive screen. Similarly, a light pen and a light sensitive screen can be used for selecting a displayed object. Such devices may thus detect selection position and the selection as a single operation instead of the "point and click," as in a system incorporating a mouse or trackball. Stylus and pen based input devices as well

as touch and light sensitive screens are well known in the art. Such a system may also lack a keyboard such as 222 wherein all interface is provided via the stylus as a writing instrument (like a pen) and the written text is interpreted using optical character recognition (OCR) techniques.

Another device which may optionally be coupled to bus 201 is a hard copy device 224 which may be used for printing instructions, data or other information on a medium such as paper, film, or similar types of media.

In one embodiment, a communication device 225 is coupled to bus 201 for use in accessing other nodes of the distributed system via a network. The communication device 225 may include any of a number of commercially available networking peripheral devices such as those used for coupling to an Ethernet, token ring, Internet, or wide area network. Note that any or all of the components of the system illustrated in **Figure 2** and associated hardware may be used in various embodiments of the present invention. However, the actual system configuration used to implement the present invention may vary from implementation to implementation.

The present invention is related to the use of computer system 200 to create and maintain replicated object groups that facilitate replication of data among nodes in a computer system. As computer system 200 executes a program, it updates a data dictionary in main memory 204 to create and maintain the replicated object groups.

REPLICATED OBJECT GROUPS

A replicated object is a set of data that is copied from one site to one or more other sites in a distributed environment. Each copy of a replicated object is referred to as a replica. Any replica of a replicated object can potentially be updated, and any updates made to one replica of a replicated object are ultimately applied at all other replicas of the replicated object. An object group is a mapping between objects in one or more schemas to logical groups. A replicated object group is an object group that is a logical collection of replicated objects. Freed of the constraints related to schemas, the members of a given replicated object group can be chosen with an eye toward facilitating data replication. The replicated objects in a given replicated object group can be chosen such that they are logically related in a way that is directly associated with replication (e.g., the replicated objects share a common replication destination). Thus, replicated object groups allow a higher level mapping which provides extra flexibility in defining groups of data that will be replicated together. Advantages of using object group-level replication include allowing objects to be distributed at a higher level of granularity, and allowing distribution of collections of replicated objects that potentially span multiple schemas.

Figure 5 is an illustration of a distributed data processing system which supports data replication according to one embodiment of the present invention. The environment for supporting data replication is referred to as the "replication environment." A replication environment includes one or more replicated object groups, the replicated objects in the replicated object groups, and the sites containing replicas of the replicated objects. The distributed data processing system of **Figure 5**

comprises node 1, node 2, and node 3 coupled together with network links 120, 125, and 130. While the system of **Figure 5** has three nodes, an arbitrary number of nodes in the distributed system may be supported in an arbitrary configuration. A first schema 110 containing objects A, B and C is resident on node 1. Also resident on node 1 is a second schema 115 containing objects D, E and F. In this example, rather than replicating data at the schema-level as in the prior approach illustrated by **Figure 1**, replicated object groups 535, 560, and 565 have been intelligently chosen to facilitate data replication. The replicated object groups in this example have been chosen to minimize the inefficiencies of propagating modifications to sites that have no need for such modifications.

Like the example illustrated in **Figure 1**, node 2 requires access to objects A, C and D, but does not require access to objects B, E or F. Node 3 requires access to objects B, C, E and F but does not require access to objects A and D. Again, dashed lines represent an operation that requires the object to be read or modified.

For the sake of illustration, assume that schema 110 contains payroll data (e.g., salary tables, accrued vacation tables, accrued sick time tables) and schema 115 contains accounting data (e.g., accounts payable tables, accounts receivable tables, and payroll tables). Further assume personnel department users are at node 2 and finance department users are at node 3. In this example, instead of copying all the objects contained in the first schema 110 and all the objects contained in the second schema 115 to node 2 and node 3, only the objects required at each node will be replicated.

The personnel department users at node 2 primarily rely on payroll data; however, they require some accounting data. In contrast, the finance department users primarily access accounting data and only some payroll data is required. Therefore, replicated object groups 560 and 565 will be copied to node 2 and replicated object groups 535 and 565 will be copied to node 3. Objects A and D are grouped together in replicated object group 560 because they are both required at node 2 by the personnel department users and neither are required by the finance department users at node 3. While the underlying data of objects A and D may be completely unrelated (e.g., accrued vacation and accounts receivable), the commonality is the fact that they will both be replicated to node 2 and not to node 3. Similarly objects B, E, and F are grouped together in replicated object group 535; their association based purely on the fact that they all will be replicated to node 3 and not to node 2.

Once the replicated object groups are copied, node 2 will contain a first schema 510 containing only objects A and C from schema 110, a second schema 515 containing only object D, a first replica 575 of replicated object group 560, and a second replica 580 corresponding to replicated object group 565. Further, node 3 will contain, a first schema 520 containing only objects B and C, a second schema 525 containing only objects E and F, a first replica 590 corresponding to replicated object group 565, and a second replica 530 corresponding to replicated object group 535.

These intelligent groupings illustrate the advantages of using replicated object groups for data replication. In contrast to the schema-based replication mechanisms, object group-based replication allows the various nodes to contain replicas of only those sets of data that are required by the nodes. Thus, in this example, the finance

department users at node 3 will not unnecessarily receive payroll data that they do not require and the personnel department users at node 2 will not be burdened with accounting data that is not required. This decreases both the storage space required for replication, and the internode traffic generated during replication. In addition, the existing schemas are not affected or restricted by the groupings specified in the replicated object groups. This example illustrates how employing replicated object groups as the logical unit of distribution facilitates data replication.

TABLE-BASED IMPLEMENTATION OF REPLICATED OBJECT GROUPS

According to one embodiment of the invention, computer system 200 uses a plurality of tables to implement replicated object groups. This plurality of tables will be discussed in detail with respect to **Figure 3**.

Figure 3 illustrates the relationship among the database data dictionary tables for one embodiment of the present invention. In **Figure 3**, one-to-many relationships are represented by the inverted V at the end of the "many" side of the one-to-many relationship.

A replicated objects table 320 is a table that contains records providing information about the objects in each replicated object group. The replicated objects table 320 lists all the members of each replicated object group. The replicated objects table 320 contains columns (attributes) indicating the characteristics and status of objects associated with each replicated object group. The columns of the replicated objects table 320 are described below in Table 1.

Replicated Objects Table

Column	Description
sname	The owner of the object.
oname	The name of the object.
type	The type of object (e.g. table, view, package, package body, procedure, function, index, synonym, trigger, or snapshot).
status	The status of the object.
id	An identifier associated with the local database object.
comment	User-defined comment for the object.
gname	The name of the replicated object group that contains the object.

Table 1

In one embodiment, the replicated objects table 320 includes an sname column. The sname column indicates the name of the schema that contains the replicated objects specified by oname. In this embodiment, the sname column identifies the user that owns the object. In another embodiment, the database has only one schema and the sname column becomes unnecessary. The oname column contains the name of the replicated object. The type column indicates the type of object that is to be replicated. Replicated objects can include database objects and structures such as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In one embodiment, the replicated objects table 320 includes a status column. The status column indicates the status of a given replica. The status column is maintained independently at the site of each replica. Exemplary status values include the following: Create - Indicates that the local database is an attempting to create the object locally; Compare - The replica is being compared with the master definition to ensure that they are consistent; Valid - The creation or

comparison has completed successfully; Error - The creation or comparison was unsuccessful. In one embodiment, the replicated objects table 320 includes an id column. The id column is a unique number that identifies the replicated object on the local node. In this embodiment, access to the object is faster and easier than accessing the object by its name (oname). The comment column allows a user supplied comment to be associated with the object. The gname column indicates the name of the replicated object group to which the object belongs. In other words, the object is a member of the replicated object group having the name indicated by the gname column. The existence of both a gname and an sname column allows a given object to be independently associated with both a schema and a replicated object group. Therefore, in an environment having multiple schemas, an advantage of this embodiment is that it allows a replicated object group to span multiple schemas. This means some of the objects in a replicated object group could be members of one schema, others could belong to a different schema, and still others might belong to a third schema. For example, in **Figure 5** replicated object group 560 contains object A from schema 110 and object D from schema 115.

In one embodiment, a replication destinations table 350 contains records providing information about the technique used to propagate data modifications and the sites to which each replicated object is replicated. The columns of the replication destinations table 350 are described below in Table 2. In this embodiment, many records in the replication destinations table 350 can be associated with a given record in the replicated objects table 320. This relationship allows changes made to a given replicated object to propagate to different sites using different propagation techniques.

For example, an object might propagate changes asynchronously to one site and data modifications to the same object could potentially propagate synchronously to another site. The columns of the replication destinations table 350 indicate the technique used to propagate operations on an object to the corresponding replica at another replication site. The operations to be propagated may have resulted from a call to a stored procedure or procedure wrapper, or may have been issued against a table directly. In one embodiment, the replication sites are classified as either snapshot sites or master sites. Master sites receive propagated changes immediately when changes are applied to any object being replicated at that site or at user-defined time-based intervals. Master sites are characterized by the fact that they can support one or more snapshot sites. A Snapshot site is refreshed from its associated master site at a time-based interval or on demand.

Replication Destinations Table

Column	Description
sname	The owner of the object.
oname	The name of the replicated object.
type	The type of object.
dblink	The fully qualified database name of the site to which changes are being propagated.
how	How the propagation is performed.
comment	User-defined comment for the object.

Table 2

The replication destinations table 350 contains an sname column, an oname column, a type column, a dblink column, a how column, and a comment column. The sname column, oname column, type column, and comment column are as described above with respect to the replicated objects table 320.

The dblink column contains the database name of a site to which data modifications will be propagated. The replication destinations table 350 will have a record corresponding to each object and each site to which the object's data modifications are propagated. For example, in **Figure 5**, the replication destinations table at node 1 will have a record indicating that node 2 is a site to which changes to object C will be propagated and the table will also have a record indicating that node 3 is a site to which changes to object C will be propagated. In this example, however, since object A is only replicated at node 2, the replication destinations table at node 1 will only need one record corresponding to object A.

The how column indicates the method of propagation to the site listed in the dblink column. Exemplary methods include synchronous and asynchronous. If the how column is synchronous, propagation to the dblink site is performed synchronously. When the how column is asynchronous, propagation to the dblink site is performed asynchronously. An advantage of providing the site and method of propagation to a given site at this level in the database design allows a given object to propagate changes to sites independently using different propagation techniques.

According to another embodiment of the invention, a generated replication support objects table 340 is provided. The generated replication support objects table 340 maintains a list of objects that have been generated to support replication. Since many support objects may be required for a given object, many records in the generated replication support objects table 340 can be associated with each record in the replicated objects table 320. The columns of the generated replication support objects table 304 are described below in Table 3.

Generated Replication Support Objects Table

Column	Description
sname	The owner of the generated object.
oname	The name of the generated object.
type	The type of the generated object.
reason	Reason for generating the object.
base_sname	Owner of the "parent" object.
base_oname	Name of the "parent" object.
base_type	Type of the "parent" object.

Table 3

The first four columns relate to the generated object. The sname column indicates the name of the schema that contains the generated object. The oname column contains the name of the generated object. The reason column indicates why the generated object was created. The type column indicates the type of the generated object. Generated objects include database objects and structures and can include triggers, packages, and procedures to support replication of objects. For example, in **Figure 5** a change to object A at node 1 might fire a generated trigger that calls a remote generated procedure at node 2 that applies the same change at the remote site to the replicated object A.

The generated replication support objects table 340 also includes a base_sname column, a base_oname column, and a base_type column. These "base" columns all refer to the "parent" object. The "parent" object is the object which the generated object was created to support. The base_sname column contains the name of the schema of which the "parent" object is a member. The base_oname column contains the name of the "parent" object. The base_type column indicates the "parent" object's type.

An object groups table 310 is a table that contains records providing information about all the replicated object groups that are being replicated. A one-to-many relationship exists between the replicated objects table 320 and the object groups table 310; therefore, allowing many objects to be added to (associated with) a given replicated object group. For example, in **Figure 5** replicated object group 560 contains objects A and D. In this example, the replicated objects table at node 1 contains a record for object A associating (registering) it with replicated object group 560 and a record for object D associating it with replicated object group 560. The object groups table 310 maintains a list of replicated object groups that have been created for replication. The columns of the object groups table 310 indicate the status of a given replicated object group. The columns of the object groups table 310 are described below in Table 4.

Object Groups Table

Column	Description
gname	The name of the replicated object group.
master	'Y' indicates that this site is a master site. 'N' indicates that this site is a snapshot site.
status	The status of the site.
comment	User-defined comment for the replicated object group.

Table 4

The object groups table 310 includes a gname column, a master column, a status column, and a comment column. The gname column contains the name associated with the replicated object group. The master column indicates whether the site where the object groups table 310 resides is a master site or a snapshot site. The

status column is available to provide further information about the site. Finally, the comment column provides the user the option of supplying a comment regarding the associated replicated object group.

In one embodiment, a schemas table 305 is provided. The schemas table 305 is a table that contains a list of schemas in the database. A one-to-many relationship exists between the replicated objects table 320 and the schemas table 305. Thus, multiple objects can be associated with each schema listed in the schemas table 305. For example, in **Figure 5** objects A, B, and C are all associated with (members of) schema 110. In another embodiment, the schemas table 305 is unnecessary because the database is the schema; therefore, only one schema would exist and all replicated objects would be associated with this schema. The columns of the schemas table 305 are described below in Table 5.

Schemas Table

Column	Description
username	The name of the user.
user_id	Unique number that identifies the user.
password	Encrypted password.
created	Date of creation.

Table 5

The schemas table 305 includes a username column, an optional user_id column, an optional password column, and an optional created column. The username column indicates the name of the user. The user_id column is not required, but may be provided for easier access to the user rather than accessing it by the

username. The password column contains a password for the associated user. The date of creation may also be associated with the user by providing a created column.

While **Figure 3** has been described with reference to tables, various alternative mechanisms may be used to represent the data dictionary entities. For example, an alternative approach in an object-oriented database would be to represent the entities 305, 310, 320, 340, and 350 as objects.

Figures 4a and **4b** illustrate a flow diagram illustrating a method of specifying a replication environment and replicating data according to one embodiment of the present invention. The method begins at step 405 where one of a plurality of sites is designated as the master definition site for the replicated object group. The master definition site is the control point for performing administrative activities for the associated replicated object group, the replicated objects in the replicated object group, and the snapshot and master sites containing replicas of the replicated object group. The master definition site is defined via a procedure call that creates a replicated object group, step 410. The replicated object group represents a mapping to one or more subsets of data within one or more sets of formatted data stored at the master definition site. The site at which the replicated object group is created is designated as the master definition site for that replicated object group. Creating a replicated object group is accomplished by inserting a record corresponding to the new replicated object group into the object groups table 310.

Once the replicated object group has been created, at step 415, replicated objects can be added to the replicated object group. Importantly, adding a replicated object to a replicated object group need not involve physically moving the data

associated with the replicated object. Rather, the data underlying the replicated object can simply be associated with the replicated object group to which it is added. Adding the replicated objects to the replicated object group registers the replicated objects and makes them eligible to be replicated to other sites. The registration is accomplished by adding a record to the replicated objects table 320. This record insertion can be performed by a procedure that is supplied with the new replicated object, for example. Usually the object to be replicated already exists at the master definition site; however, in one embodiment, the procedure that performs the registration also accepts one or more Structured Query Language (SQL) Data Definition Language (DDL) statements which create the replicated object as well.

In a replication configuration where one or more replication sites will propagate changes asynchronously, it is possible for conflicting updates to occur. For example, a conflict can be caused by two replication sites modifying the same replicated object before propagating their updates to each other. In one embodiment, if conflict resolution routines are desired they can be supplied via a procedure call, step 430. Many conflict resolution methods are available including: applying the data with the latest timestamp, applying all data additively, and applying the value from the replication site with the highest priority.

In another embodiment, replication support objects can be generated to support data replication, step 435. Replication support objects include database object such as triggers, packages and procedures. Step 435 further includes adding entries to the generated replication support objects table 340 for each of the generated objects.

The next step is to determine if more replicated objects are to be replicated with

this replicated object group, step 440. If more replicated objects are to be added to the replicated object group, then steps 415 through 435 can be repeated until all the replicated objects that can be grouped together for replication have been added to the replicated object group. Otherwise, when the replicated object group has been completely defined and all the objects have been added, processing proceeds to step 450.

At step 450, user input is received that defines one or more sites as replication sites. This information will ultimately be written to the object groups table at the corresponding remote sites.

At step 455, the member replicated objects of the replicated object group are created at each remote replication site. The data associated with the replicated objects is replicated to each of the remote replication sites. At this point, the replication environment has been configured.

Once the replicated environment has been configured, the flow continues to step 460. At step 460, no action is taken until a data modification to a replicated object in the replicated object group is detected.

When a data modification is detected to one of the members of the replicated object group, processing proceeds to step 470. At step 470, any changes made to one of the replicated objects in the replicated object group are propagated to the replication sites. The changes are propagated to all master sites immediately or at a predetermined time-based interval. The master sites propagate the changes to associated snapshot sites at a predetermined time-based interval or on demand. Steps 460 and 470 are repeated for each data modification to a replicated object in the replicated object group.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

Diane D. Mizrahi October 16, 1997

CUSTOMER REQUEST SUMMARY

Your request was:

=e003
=
=Word frequency list for document 667677
=
=---search-id---
=667677, mizrahi diane
=---search-id---
=
=---word freq---
= 1 ability 1 abstract
= 1 acce 14 access
> 1 according 1 activities
> 1 acts 1 administer
> 1 administering 1 administrator
> 1 advantage 1 aintained
> 8 all 5 allow
> 1 allowing 4 also
> 1 amount 33 and
> 5 another 4 apparatus
> 1 apparent 1 application
> 5 approach 1 arbitrary
> 6 are 1 arise
> 1 aspect 1 assign
> 1 assigned 1 assuming
> 2 assumption 1 available
> 1 bandwidth 1 barriers
> 1 become 2 been
> 4 between 1 boundaries
> 2 breaking 1 burden
> 3 but 8 can
> 1 case 1 certain
> 1 circumstances 1 complexity
> 1 composed 3 comprise
> 1 comprises 2 computer
> 1 consequences 1 considers
> 1 constraints 5 contain
> 3 containing 2 content
> 1 controls 1 copied
> 2 copies 1 copy
> 1 corresponding 2 could
> 1 coupled 4 created
> 3 creating 1 dashed
>56 data 10 database
> 1 dba 1 define
> 1 defined 1 defining
> 1 depends 1 described
> 2 desirable 1 destroy
> 2 detected 3 different
> 1 disadvantage 2 disadvantages
> 1 disclosed 1 disclosure
> 2 discussed 1 distribute

> 8 distributed	1 distribution
> 1 division	4 does
> 1 down	1 dramatic
> 1 due	11 each
> 2 efficiency	1 efficiently
> 1 either	1 employing
> 1 employs	1 eplication
> 1 erfon	1 established
> 1 even	2 every
> 9 example	1 excess
> 2 exist	1 faced
> 1 facilitating	1 faster
> 1 feasible	1 field
> 3 figure	1 filed
> 1 finer	12 first
> 1 following	26 for
> 9 formatted	1 from
> 2 further	1 generally
> 1 generated	1 generation
> 2 given	3 granularity
> 7 group	1 groupings
>10 groups	5 has
> 6 have	2 having
> 1 higher	1 hold
> 1 however	1 illustrated
> 1 illustrates	1 impose
> 2 includes	1 increase
> 2 increased	1 increases
> 2 independent	1 individual
> 1 inefficiencies	1 information
> 1 initially	1 interconnected
> 1 into	1 inve
> 3 invention	1 iocal
> 1 jain	1 june
> 1 leads	1 less
> 9 level	1 levels
> 1 likewise	1 limitations
> 1 lines	1 links
> 1 little	5 local
> 2 logical	1 logically
> 1 lower	1 lowest
> 1 maintain	1 maintaining
> 4 management	1 manner
> 4 mapped	5 mapping
> 1 mappings	1 matted
> 1 may	3 mechanism
> 1 member	1 members
> 1 mere	7 method
> 1 modem	5 modification
> 3 modifications	1 modified
>16 more	1 muc
> 5 multiple	5 must
> 1 names	1 nany
> 1 necessary	1 need
> 2 needs	2 network
> 1 ning	34 node
> 6 nodes	6 not
> 1 ntion	3 number

>19 object	23 objects
>23 one	6 only
> 1 operation	1 operations
> 1 organized	3 other
> 1 over	1 participating
> 1 partitioned	1 passed
> 1 patent	3 perform
> 1 performed	1 performing
> 5 physical	2 plurality
> 1 pose	1 potentially
> 1 power	1 presence
> 1 previously	2 prior
> 1 problem	5 processing
> 2 proliferation	2 propagated
> 3 provide	2 provided
> 1 purpose	1 read
> 2 receive	1 recognize
> 1 recognizes	1 reduces
> 2 referred	1 referring
> 1 regarding	1 regardless
> 2 registered	2 related
> 2 relates	1 relative
> 3 remote	7 replica
> 1 replicafiori	2 replicas
> 1 replicate	13 replicated
> 4 replicating	26 replication
> 1 represent	1 represented
> 1 represents	4 require
> 3 required	4 requires
> 1 rernote	2 resident
> 2 resides	2 residing
> 2 respect	1 result
> 1 risk	1 said
> 6 same	1 sandeep
>30 schema	1 schemallo
>18 schemas	8 second
> 4 security	1 separate
> 4 set	6 sets
> 3 several	1 siniilarly
> 5 site	8 sites
> 1 solution	1 solve
> 2 some	1 span
> 1 specifically	5 specified
> 1 step	1 storage
> 2 store	5 stored
> 1 structure	7 subset
> 4 subsets	1 summary
> 1 sys	12 system
> 4 systems	1 take
> 1 tems	7 than
>17 that	94 the
> 1 thedataspecifiedintheseco	1 their
> 1 then	1 therefore
> 4 these	12 this
> 1 thod	1 thought
> 1 titled	1 together
> 4 traffic	1 true
> 1 two	1 tying

> 1 type 1 typically
> 2 underlying 3 unit
> 1 unnecessarily 2 unnecessary
> 4 upon 1 used
> 3 user 2 users
> 3 using 1 utilize
> 1 way 1 were
> 5 when 1 whether
> 4 which 11 will
> 1 willhavetobe 8 with
> 3 within 1 words
> 6 would 1 ysterns
>
>---word freq---
>
>---number returned---
>50
>---number returned---
>
>---output options---
>abstracts
>exemplary claims
>field of search 10
>titles
>---output options---
>

Sales Order Summary:
Customer ID: 681
Sales Transaction Nbr: 69109
Date Posted: October 9, 1997
Product: E003
Quantity: 50

E003 WORD FREQUENCY SEARCH REPORT

Classification Analysis:

1. 364/DIG. 1 Total=38 ORs=0 XRs=38
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub DIG. 1 GENERAL PURPOSE PROGRAMMABLE DIGITAL COMPUTER SYSTEMS

2. 364/282.1 Total=15 ORs=0 XRs=15
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.15 SYSTEM MANAGEMENT (SOFTWARE)
Sub 282.1 .Data base

3. 364/DIG. 2 Total=14 ORs=0 XRs=14
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub DIG. 2 GENERAL PURPOSE PROGRAMMABLE DIGITAL COMPUTER SYSTEMS

4. 364/242.94 Total=7 ORs=0 XRs=7
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.6 INTERFACING OR COMMUNICATION TECHNIQUE
Sub 242.94 .Networking

5. 364/282.4 Total=7 ORs=0 XRs=7
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.15 SYSTEM MANAGEMENT (SOFTWARE)
Sub 282.1 .Data base
Sub 282.4 ..Distributed

6. 364/286.4 Total=6 ORs=0 XRs=6
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.15 SYSTEM MANAGEMENT (SOFTWARE)
Sub 286 .Program management
Sub 286.4 ..Security

7. 364/286.5 Total=6 ORs=0 XRs=6
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.15 SYSTEM MANAGEMENT (SOFTWARE)
Sub 286 .Program management
Sub 286.4 ..Security
Sub 286.5 ...Access/authentication

8. 364/228.1 Total=5 ORs=0 XRs=5
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.3 SYSTEM ARCHITECTURE
Sub 228.1 .Shared memory

9. 364/230 Total=5 ORs=0 XRs=5
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.3 SYSTEM ARCHITECTURE
Sub 230 .Multiprocessor/processor control

10. 364/231.4 Total=5 ORs=0 XRs=5
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.3 SYSTEM ARCHITECTURE
Sub 231.4 .Timeshared

11. 364/231.6 Total=5 ORs=0 XRs=5
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.3 SYSTEM ARCHITECTURE
Sub 231.4 .Timeshared
Sub 231.6 ..Plural programs (multiprogrammed)

12. 364/240 Total=5 ORs=0 XRs=5
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.6 INTERFACING OR COMMUNICATION TECHNIQUE
Sub 240 .Bus

13. 364/242.95 Total=5 ORs=0 XRs=5
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.6 INTERFACING OR COMMUNICATION TECHNIQUE
Sub 242.94 .Networking
Sub 242.95 ..Local area network (LAN)

14. 364/286 Total=5 ORs=0 XRs=5
Class 364 ELECTRICAL COMPUTERS AND DATA PROCESSING SYSTEMS
Sub ???.15 SYSTEM MANAGEMENT (SOFTWARE)
Sub 286 .Program management

Patent Report:

Ref	Patent Id	Issue/File	US Class (OR)	Title
1	05560005	Sep 24 1996 Feb 25 1994	395/610	Methods and systems for object-based relational distributed databases

Inventor: Hoover; Michael K., et al.
Assignee: ActaMed Corp.
Abstract:

An object-based relational distributed database system and associated methods of operation that transforms data stored in a plurality of remote, heterogeneous user databases into a homogeneous data model is disclosed. Data stored in distributed, heterogeneous user database structures is homogenized by mapping into object attributes of predetermined instances of objects forming to a conceptual model that relates the various heterogeneous databases. The object attributes are stored in remote databases at client sites, which can be separate computer systems from the heterogeneous user databases or separate processes running on a computer system that maintains the heterogeneous user databases. The system stores location information and status information relating to the homogenized data in a centralized object broker for object management, thereby facilitating location and retrieval of data items from one or more of the remote, heterogeneous user databases.

Exemplary Claim [1]:

1. A method of operating a distributed data processing system including a plurality of independent remotely located user computers that process user data in user databases and at least one object broker computer,

the user computers being interconnected with the object broker computer by data communication hardware over a data communication network,

the user computers being operative to perform data operations of storing, updating, and retrieving user data items in response to user commands, comprising the steps of:

(a) for a subject at one of the user computers for which data is to be processed in the system, creating an object instance by assigning a unique object identifier to data items associated with the subject by:

(a1) providing a global object identification address space corresponding to a range of object identifiers for association with a plurality of subjects; and

(a2) allocating a predetermined range of object identifiers within the global object identification address space to each remotely located user computer;

(b) storing the data items associated with the subject at the user computer in association with the object identifier;

(c) at the object broker computer, storing the locations of the user computers in a mapping table in association with object identifiers;

(d) in response to a query to the object broker computer for data relating to a particular subject in question, retrieving a selected object identifier for the subject in question;

(e) in response to retrieval of an object identifier for the subject in question in the preceding step, retrieving the location of a remote user computer associated with the selected object identifier; and

(f) retrieving data stored at the remote user computer associated with the selected object identifier via the data communication network.

2	04432057	Feb 14 1984 Nov 27 1981	395/608	Method for the dynamic replication of data under distributed system control to control utilization of resources in a multiprocessing,

distributed data base system

Inventor: Daniell; Thomas P. et al.
Assignee: International Business Machines Corporation
Abstract:

A method for dynamic replication of data under distributed system control to control the utilization of resources in a multiprocessing, distributed data base system. Previously, systems providing for data replication at nodes of a multiprocessing, distributed data base system required that a central node maintain control, or that replicated data be synchronized by immediately conforming all copies of an updated data item. By this invention, requests for access to data of a specified currency are permitted and conformation of updated data is selectively deferred by use of a control procedure implemented at each node and utilizing a status and control (SAC) filed at each node which describes that node's view of the status for shared data items at other nodes.

Exemplary Claim [1]:

1. A method for operating a computing apparatus having a plurality of nodes, each node having storage means for storing a plurality of data items and means responsive to a request for accessing a specified data item, and communication means interconnecting selected nodes, the method including the steps of controlling access to the specified data item and controlling the modification of copies of the data item, characterized by the steps of:

distributing data access control to each node of the apparatus; and, responsive to a request having a specified currency, dynamically replicating data items while selectively deferring conformation of the replicated data;

such that the most current data items migrate to their respective affinity nodes.

3 05627961 May 6 1997 395/182.04 Distributed data processing system
Jan 23 1996

Inventor: Sharman; Geoffrey
Assignee: International Business Machines Corporation
Abstract:

Distributed data processing systems with replication of data across the system are required to provide data access and availability performance approaching that of a stand-alone system with its own data, but reductions in network traffic gained by replication must be balanced against the additional network traffic required to update multiple copies of the data. Apparatus and a method of operating a distributed data processing system is provided in which a currency period is associated with each copy of a data object and the data object is assumed to be valid during the currency period. The apparatus has means for checking (500) whether the currency period has expired and means for updating (570, 580) the copies. A validity indicator is set on determination of expiry of the currency period and when updates are applied to the primary copy, and is checked to determine validity of a copy.

Exemplary Claim [1]:

1. In a distributed data processing system comprising a plurality of data processors in communication with each other and having a central processing unit (CPU) and memory, a method of distributed data processing comprising the steps of:

partitioning a data record into non-overlapping parts;

distributing the parts of the data record across a plurality of data processors such that a first data processor holds a primary copy of a part of the data record;

storing at one or more remote data processors (200, 250) a replicated

secondary copy (310) of the primary copy of the part of the data record stored at the first data processor (100);

propagating (420, 570) updates of said primary copy to the remote data processors (200, 250);

storing at each of the remote data processors (200,250) a currency expiry time (320) associated with each secondary copy stored at a remote data processor, wherein when an update is propagated to the remote data processor, the associated currency expiry time is updated;

in response to a request for data access at a remote data processor having a secondary copy, checking (510) whether the currency expiry time (320) has passed for the secondary copy of the data.

4 04714992 Dec 22 1987 395/622 Communication for version management in a distributed information service
Nov 26 1985

Inventor: Gladney; Henry M. et al.
Assignee: International Business Machines Corporation
Abstract:

In a distributed processing system network in which at least one node operates as a source location having access to data objects of a database, and at least one other node operates as a replica location storing replicas of data objects from the source location, managing obsolescence of the replicas is performed by having the replica locations submitting requests to the source location for ascertaining obsolescence of data objects. The source location, responsive to a request from a requesting replica location, extracts identifiers of a set of obsolete objects and communicates them to the requesting replica location. Upon receiving the identifiers, the requesting location renders inaccessible those data objects corresponding to the identifiers received. The source location then removes those identifiers that have been communicated to the requesting replica location.

Exemplary Claim [1]:

1. A method for managing obsolescence of replicas of data objects, the objects being utilized in multiple nodes of a distributed processing system in which at least one node operates as an object source location having access to a source database containing source data objects and at least one other node operates as an object replica location having means for storing replicas of requested objects received from a source location, each source data object being alterable whereby replicas of altered objects stored at replica locations may become obsolete, comprising the steps of:

responsive to a request from a first replica location to ascertain obsolescence of data objects, extracting at the source location identifiers of a set of obsolete objects;

communicating said identifiers, if any, as an atomic demand/response transaction to said first replica location;

rendering inaccessible at said first replica location any replicas corresponding to those identifiers received from the source location; and

removing from the source location those identifiers communicated to said first replica location.

5 05586310 Dec 17 1996 395/610 System for distributed database replicated read with exclusive central server transfer of primary copies
Nov 23 1993

Inventor: Sharman; Geoffrey
Assignee: International Business Machines Corporation
Abstract:

When an update is made to a data record in a distributed, replicated data processing system, the update is first applied to a primary copy of the data record before being applied to any other copy to ensure that updates are applied in the correct time sequence. Apparatus and a method of operating a distributed data processing system is provided in which responsibility for the primary copy is transferable to whichever processor in the system requires most frequent update access, providing improved performance and availability of data. The primary copy may be partitioned and distributed across the system.

Exemplary Claim [1]:

1. A method of distributed data processing comprising the steps of:

storing replicated copies of a data record (310) at a plurality of data processors (100, 200, 250);

designating a first one of said copies at a first one of said data processors as the primary copy for update purposes and the other copies as secondary copies:

applying updates to the data record firstly to the primary copy and subsequently propagating (420, 570) updates of said primary copy to the other data processors holding secondary copies of the data record;

in response to a request (800) from one of said data processors checking (820) whether a central data processor has responsibility for the primary copy or responsibility has been transferred to a second one of said data processors; and

only if the central processor has said responsibility transferring (850) responsibility for the primary copy from said first data processor to said requesting data processor wherein the secondary copy of the data record at the requesting processor is then designated as the primary copy of the data record for update purposes such that future updates to the data record will firstly be applied at the requesting processor and will subsequently be propagated to the other data processors.

6 05598566 Jan 28 1997 395/750.06 Networked facilities management
Jan 7 1994 system having a node configured with distributed load management software to manipulate loads controlled by other nodes

Inventor: Pascucci; Gregory A. et al.
Assignee: Johnson Service Company
Abstract:

A networked system having a wide variety of applications and particularly applicable to facilities management systems has multiple levels of software in processing nodes. The levels include a "features" processing level which communicates requests for data to a software object level containing databases of processes and attributes and database managers. The database managers in the software object level operate to provide data to the high level features in the same format. The software object level communicates with a hardware object level which also contains databases and database managers to mask differences between operational hardware units. By categorizing operational units by type, additional units of a known type can be added with only low level hardware object database changes. Adding units of a new type is facilitated by software changes confined to the lower level hardware and software objects, avoiding software changes at high level features. Individual software objects are tailored for typical types of inputs and output devices encountered by facilities management systems. Universal drive circuitry also provides applicability to a broad range of devices. Nodes store restoration characteristics of loads controlled by the nodes. After being shed to limit energy consumption, software local to the node restores the

loads under predefined circumstances independently of the load shed commands to achieve distributed load management.

Exemplary Claim [1]:

1. A method of limiting energy consumption of a network having loads controlled by nodes communicating over the network, the nodes having storage means and processing means, the nodes including a master node having a high level load shedding software feature and other nodes having local software object features, the local software object features controlling the loads, the method comprising steps of:

storing load restoration characteristics of the loads controlled by the other nodes in the storage means of the other nodes;

executing the high level load shedding software feature in the processing means of the master node to limit energy consumption of the network, and subsequently transmitting over the network commands to shed particular loads controlled by the other nodes;

executing predefined load shedding processing in the other nodes controlling the particular loads by using the local software object features, each of the local software object features having a data base manager and attributes stored in a data base in each of the other nodes, the local software object features shedding the particular loads in response to the commands;

under control of the local software object features, restoring the particular loads independently of the commands from the master node in response to the attributes of the local software object features.

7 05444851 Aug 22 1995 395/200.52 Method of accessing configured nodes in a facilities management system with a non-configured device
Jan 21 1994

Inventor: Woest; Karen L.

Assignee: Johnson Service Company

Abstract:

A networked system having a wide variety of applications and particularly applicable to facilities management systems has multiple levels of software in processing nodes. The levels include a "features" processing level which communicates requests for data to a software object level containing databases of processes and attributes and database managers. The database managers in the software object level operate to provide data to the high level features in the same format. The software object level communicates with a hardware object level which also contains databases and database managers to mask differences between operational hardware units. By categorizing operational units by type, additional units of a known type can be added with only low level hardware object database changes. Adding units of a new type is facilitated by software changes confined to the lower level hardware and software objects, avoiding software changes at high level features. Individual software objects are tailored for typical types of inputs and output devices encountered by facilities management systems. Universal drive circuitry also provides applicability to a broad range of devices. A non-configured node is connected to a configured network at a location defined by a subnet and a local address. The location information and a drop identifier are used as a network address for the non-configured node to transmit and receive messages. Routing information stored in the configured nodes allows non-configured node to communicate with other network nodes.

Exemplary Claim [1]:

1. A method of accessing network controllers in a facilities management system with a portable computing unit, the network controllers in the facilities management system being arranged to control a process,

the network controllers being configured as at least one network and being interconnected by at least one communication link, each of the network controllers including an equipment interface for receiving data related to the process, and a processor including a drop port, the processor being coupled to the equipment interface, the facilities management system being initialized so that the network controllers are configured to each have a network address indicative of a particular location in the facilities management system, the network address including a subset indicative of an associated communication link to which the network controller is connected, a local address indicative of the network controller, and a node drop ID indicating that the network controller is a configured network controller, the portable computing unit being a non-configured device not part of the network, the method comprising:

connecting the portable computing unit to a first port of a first network controller of the network controllers, the first configured network controller configured on the system at a first location defined by the subset indicative of the communication link and a first local address indicative of the first network controller;

assigning a first network address to the portable computing unit, the first network address including the subset, the first local address and a first drop identifier indicative of the first drop port of the first network controller;

transmitting a request for data received at an equipment interface of a second network controller located at a second network address from the portable computing unit to the second network controller, the request including the second network address as a destination indicator and the first network address as a source indicator;

transmitting the data from the second network controller to the portable computing unit in response to the request for data, the data including the second network address as the source indicator and the first network address as the destination indicator;

receiving the data from the second network controller at the processor of the first network controller according to the subset and local address of the first network address;

transmitting the data to the portable computing unit through the first drop port specified by the first drop identifier portion of the first network address.

8 05463735 Oct 31 1995 395/200.52 Method of downloading information
Feb 3 1994 stored in an arching device to destination network controller through intermediate network controllers in accordance with routing information

Inventor: Pascucci; Gregory A. et al.
Assignee: Johnson Service Company
Abstract:

A network system having a wide variety of applications and particularly applicable to facilities management systems includes network controllers which continuously process data related to building and industrial, environmental, security and other automated system controls. Each network controller has a network address indicative of a communication link to which the network controller is connected, a local address and a node drop ID to determine whether the network controller is a configured or non-configured device. Data stored in an archive device is downloaded to a destination network controller in the absence of a routing table in the destination network controller by transmitting a download request message from the archive device to an intermediate network

controller with a routing table. The intermediate network controller assumes control of the download request by transmitting the message to the destination controller. The destination controller acknowledges receipt of the message by transmitting an acknowledge message back to the intermediate network controller, which passes the acknowledge message to the archive device in accordance with the routing information stored in the intermediate network controller. Thus, as certain network controllers are connected, disconnected or disabled during the operation of the network, the control of a process is not interrupted. Additionally, the network controllers are not configured to store large amounts of routing data because a path to a device can be established through other controllers with routing information.

Exemplary Claim [1]:

1. A method of downloading information stored in an archive device to a destination network controller in a facilities management system including a plurality of network controllers and an archive device having device non-volatile memory, the network controllers being arranged to control a process, the network controllers being arranged as at least one network and being interconnected by at least one communication link, each of the network controllers including a controller non-volatile memory, an equipment interface for receiving data relating to the process, and a processor including a drop port, the processor being coupled to the equipment interface, the facilities management system being initialized so that the network controllers are configured to each have a network address indicative of a particular location in the facilities management system, the network address including a subset indicative of an associated communication link to which the network controller is connected, a local address indicative of the network controller, and a node drop ID indicating whether the network controller is a configured or a non-configured network controller, the network address being stored in the non-volatile memory of the network controller, the method comprising the steps of:

storing in the non-volatile memory of the archive device a destination network address of the destination network controller and an intermediate address of an intermediate network controller of the plurality of network controllers the destination address being indicative of the location of the destination network controller, the intermediate address being indicative of the location of the intermediate controller;

transmitting at least one message from the archive device to the intermediate network controller of the plurality of network controllers, the intermediate network controller having a routing table including routing information;

transferring control of transmission of the message to the intermediate network controller and passing the message from the intermediate network controller to the destination network controller in accordance with the routing information;

determining if the message should be forwarded from the destination network controller to a non-configured destination network controller coupled to the destination network controller in accordance with the node drop ID;

acknowledging receipt of the message by transmitting an acknowledge message from the destination network controller to the intermediate network controller; and

transferring control of transmission of the acknowledge message to the intermediate network controller, and passing the acknowledge message from the intermediate network controller to the archive device in accordance with the routing information.

9 05550980 Aug 27 1996 359/111 Networked facilities management system with optical coupling of local network devices
Jan 7 1994

Inventor: Pascucci; Gregory A. et al.
Assignee: Johnson Service Company
Abstract:

A networked system having a wide variety of applications and particularly applicable to facilities management systems has multiple levels of software in processing nodes. The levels include a "features" processing level which communicates requests for data to a software object level containing databases of processes and attributes and database managers. The database managers in the software object level operate to provide data to the high level features in the same format. The software object level communicates with a hardware object level which also contains databases and database managers to mask differences between operational hardware units. By categorizing operational units by type, additional units of a known type can be added with only low level hardware object database changes. Adding units of a new type is facilitated by software changes confined to the lower level hardware and software objects, avoiding software changes at high level features. Individual software objects are tailored for typical types of inputs and output devices encountered by facilities management systems. Universal drive circuitry also provides applicability to a broad range of devices. An optical isolator is connected between signal line outputs from a node and a line driver connected to a pair of lines on a bus. Similarly, an optical isolator is connected between signal line receiving inputs and a line receiver. The signal lines can be biased to predetermined levels to reduce node sensitivity to noise.

Exemplary Claim [1]:

1. A computerized node controlling at least one slave device connected to a slave device bus having a pair of signal lines, the computerized node communicating with the slave devices over the slave device bus and being optically isolated from the slave device bus, the node having a mode output for providing a mode select signal, a transmit output and a receive input, the node comprising:

a transmit optical isolator connected between the transmit output of the node and a line driver connected to the pair of signal lines;

a receive optical isolator connected between the receive input of the node and a line receiver connected to the pair of signal lines; and

a mode optical isolator having a mode input coupled to the mode output, the mode optical isolator activating at least one of the line driver or the line receiver in response to the mode select signal, the mode select signal being indicative of a transmit mode or a receive mode, the pair of signal lines receiving signals from the slave device bus in the receive mode and transmitting signals to the slave device bus in the transmit mode.

10 05511188 Apr 23 1996 395/619 Networked facilities management system with time stamp comparison for data base updates
Dec 30 1993

Inventor: Pascucci; Gregory A. et al.
Assignee: Johnson Service Company
Abstract:

A networked system having a wide variety of applications and particularly applicable to facilities management systems has multiple levels of software in processing nodes. The levels include a "features" processing level which communicates requests for data to a software object level containing databases of processes and attributes and database

managers. The database managers in the software object level operate to provide data to the high level features in the same format. The software object level communicates with a hardware object level which also contains databases and database managers to mask differences between operational hardware units. By categorizing operational units by type, additional units of a known type can be added with only low level hardware object database changes. Adding units of a new type is facilitated by software changes confined to the lower level hardware and software objects, avoiding software changes at high level features. Individual software objects are tailored for typical types of inputs and output devices encountered by facilities management systems. Universal drive circuitry also provides applicability to a broad range of devices. In each node a time stamp indicates the most recent update of the node's data base. Periodically each node transmits its time stamp. When a node receives a time stamp later than its own, the receiving node requests the transmitting node to transmit its data base to update the receiving node with the earlier time stamp.

Exemplary Claim [1]:

1. A method of synchronizing a plurality of data bases stored in a network, the network including a plurality of nodes communicating over at least one communication link, each of the nodes including a storage means for storing at least one of the data bases, the method comprising steps of:

storing in a first storage means of a first node of the nodes a first time stamp indicating a most recent time of updating a first data base stored in the first node;

automatically and periodically transmitting from the first node the first time stamp stored in the first node;

receiving a second time stamp from a second node of the nodes and comparing the second time stamp with the first time stamp, the second time stamp being stored in a second storage means of the second node and indicating a most recent time of updating a second data base stored in the second node; and

automatically requesting the second node to transmit the second data base stored in the second node to the first node if the second time stamp is later than the first time stamp.

11 05522044 May 28 1996 395/200.52 Networked facilities management
Jan 21 1994 system

Inventor: Pascucci; Gregory A. et al.

Assignee: Johnson Service Company

Abstract:

A networked system having a wide variety of applications and particularly applicable to facilities management systems has multiple levels of software in processing nodes. The levels include a "features" processing level which communicates requests for data to a software object level containing databases of processes and attributes and database managers. The database managers in the software object level operate to provide data to the high level features in the same format. The software object level communicates with a hardware object level which also contains databases and database managers to mask differences between operational hardware units. By categorizing operational units by type, additional units of a known type can be added with only low level hardware object database changes. Adding units of a new type is facilitated by software changes confined to the lower level hardware and software objects, avoiding software changes at high level features. Individual software objects are tailored for typical types of inputs and output devices encountered by facilities management systems. Universal drive circuitry

also provides applicability to a broad range of devices. Nodes are provided with ports which may accommodate a non-configured device. When a non-configured device is identified on a port, a drop identifier specifying the port is transmitted to the non-configured device for use in establishing a network address by which the non-configured device can be accessed.

Exemplary Claim [1]:

1. A facilities management system configured to allow access to the system by a non-configured portable computing unit, the facilities management system including a plurality of network controllers arranged to control a process, the network controllers being configured as at least one network and being interconnected by at least one communication link, each of the network controllers including an equipment interface for receiving data related to the process, and a processor including a drop port, the processor being coupled to the equipment interface, the facilities management system being initialized so that the network controllers are configured to each have a network address indicative of a particular location in the facilities management system, the network address including a subset indicative of an associated communication link to which the network controller is connected, a local address indicative of the network controller, and a node drop ID indicating that the network controller is a configured network controller, the facilities management system comprising:

a first configured network controller including a first processor having a first port for receiving the portable computing unit, the first configured network controller configured on the system at a first location defined by a first subset indicative of the communication link, a first local address indicative of the first configured network controller and the node drop ID;

a second network controller having a first equipment interface, the second network controller being coupled to the communication link and being configured on the system at a second location defined by a second subset indicative of the communication link and a second local address indicative of the second configured networked controller, the second network controller having a second network address including the second subset, the second local address and the node drop ID;

means for assigning a first network address to the portable computing unit, the first network address including the first subset, the first local address and a first drop identifier indicative of the first port;

means for transmitting a request for data received at the first equipment interface of the second configured network controller located at the second location from the portable computing unit to the second network controller, the request including the second network address as a destination indicator and the first network address as a source indicator;

means for transmitting the data from the second configured network controller to the portable computing unit in response to the request for data, the data including the second network address as the source indicator and the first network address as the destination indicator;

means for receiving the data from the second configured network controller at the first processor of the first network controller according to the subset and local address of the first configured network address; and

means for transmitting the data to the portable computing unit through the first drop port specified by the first drop identifier.

12 05384697 Jan 24 1995 364/139 Networked facilities management
Dec 30 1993 system with balanced differential
analog control outputs

Inventor: Pascucci; Gregory A.
Assignee: Johnson Service Company
Abstract:

A networked system having a wide variety of applications and particularly applicable to facilities management systems has multiple levels of software in processing nodes. The levels include a "features" processing level which communicates requests for data to a software object level containing databases of processes and attributes and database managers. The database managers in the software object level operate to provide data to the high level features in the same format. The software object level communicates with a hardware object level which also contains databases and database managers to mask differences between operational hardware units. By categorizing operational units by type, additional units of a known type can be added with only low level hardware object database changes. Adding units of a new type is facilitated by software changes confined to the lower level hardware and software objects, avoiding software changes at high level features. Individual software objects are tailored for typical types of inputs and output devices encountered by facilities management systems. Universal drive circuitry also provides applicability to a broad range of devices. The sum of an analog input signal and a feedback signal is provided to a buffer circuit. A current sensing network connected to the output of the buffer forms a first control signal. An equal amplitude, oppositely polarized signal forms a second control signal. An external signal switches the feedback network to produce a voltage derived from a current output of the first control signal and a voltage output of the first control signal.

Exemplary Claim [1]:

1. Apparatus for generating first and second balanced differential analog control signals comprising:

a) means for summing an analog input signal and an output from a feedback network connected between one of said control signals and said summing means;

b) a buffer circuit having an input connected to an output of said summing means and an output connected to a current sensing network input, said current sensing network having an output being said first control signal;

c) means connected to said current sensing network output for generating a balanced signal having a same amplitude and an oppositely polarized sense from said first control signal, said balanced signal being said second control signal; and

d) means in said feedback network, and activated by an external signal, for switching said feedback network output between a voltage derived from a voltage output of said first control signal and a voltage derived from a current output of said first control signal.

13 04714996 Dec 22 1987 395/619 Impact calculation for version management in a distributed information service
Nov 26 1985

Inventor: Gladney; Henry M. et al.
Assignee: International Business Machines Corporation
Abstract:

In a distributed processing system having a source node accessing data objects from a database and a replica node storing replicas of requested source data objects received from the source location, the impact to replicas caused by a change in a source data object is calculated by assigning a version number to the change. An identifier of the portion of the source database affected by the change is generated, as well as a list of replicas containing objects from the affected portion of the database.

For a replica location communicating with the source location, a table of the replicas from the list is then recorded along with the version number for communicating to the replica location.

Exemplary Claim [1]:

1. In a system for managing obsolescence of replicas of data objects, the objects being utilized in multiple nodes of a distributed processing system having at least one node operating as a source location having access to a source database containing data objects and at least one node operating as an object replica location having means for storing replicas of requested data objects received from a source location, each source data object being alterable whereby a change in a source data object will cause an impact to its replicas, a method for calculating the impact of a change to a source data object within the system, comprising the steps at the source location of:

(a) responsive to the change in the source data object, assigning a version number to the change and generating an identifier of a portion of the source database affected by the change;

(b) responsive to said identifier, generating a list of replicas including objects from the affected portion of the source database; and

(c) for a replica location in communication with the source location, recording a bad messages table of replicas in the list along with the version number assigned to the change for communication to the replica location.

14 05274789 Dec 28 1993 395/416 Multiprocessor system having distributed shared resources and dynamic and selective global data replication
Feb 25 1991

Inventor: Costa; Maria et al.

Assignee: Bull HN Information Systems Italia S.p.A.

Abstract:

Multiprocessor system having distributed shared resources and dynamical and selective global data replication in which a plurality of processors communicate each with the other through a sytem bus. Each CPU is provided with a local memory storing data used locally and global data shareable by a plurality of processes operative in differing CPUs and therefore replicated in the local memory of each CPU. The global data replication is performed, at page level, only when a global data page is effectively needed by a plurality of processes operative in differing CPUs and in those CPUs where the page is needed, the replication in the other CPUs being performed in a predetermined trash page of the local memory so that memory space required for replication is minimized, as is traffic on the system bus for global data replication and global data writes required for assuring global data consistency.

Exemplary Claim [1]:

1. A multiprocessor system having distributed shared resources and dynamic and selective global data replication, comprising:

a system bus, and

a plurality of CPUs, each CPU comprising:

a processor,

a local memory,

a memory management unit (MMU) coupled to said processor for converting a virtual information page address generated by said processor into an address having a bit field which characterizes the address as either real and related to a page of global data which needs to be replicated to the local memory of all said CPUs or physical and related to a page of local data,

an interface unit connected to said system bus and enabling the

exchange of information among said plurality of CPUs through said system bus,

an address translation unit coupled to said local memory for converting a real address, received from either said MMU or said system bus, through said interface unit, into a physical address referencing an entry of said local memory,

a memory table (GSD) replicated in the local memory of each of said plurality of CPUs, and storing information which, for each address in a set of virtual page addresses;

defines the related data page as a global data page or a local data page,

specifies in which of said CPUs the data page is valid, and specifies the real address to be used for referencing said data page,

a trash page in said local memory of each of said plurality of CPUs, each said trash page having a predetermined physical address, said trash page having a predetermined physical address, said address translation unit, in each of said CPUs, being initialized for converting a set of said real addresses into said predetermined trash page physical addresses,

first logic means, said first logic means in a first CPU among said plurality of CPUs adapted to perform a page fault procedure in response to an indication of missing page generated by the said first CPU's MMU, and referencing said GSD table in said first CPU's local memory to detect if said missing page is valid in another CPU among said plurality of CPUs and stored in said another CPU's local memory and, when said missing page is found to be valid in a second CPU;

allocating a physical page of said first CPU's local memory for storing said missing page,

modifying the contents of said first CPU's address translation unit to reference said allocated physical page when a predetermined real address is used to reference said missing page, and

requesting said second CPU to replicate said page, valid in said second CPU, as a global data page at said predetermined real address, and

second logic means, said second logic means in said second CPU being responsive to said request from said first, CPU to copy said page valid in said second CPU at said predetermined real address as a global data page,

whereby said page is replicated in a physical page other than said trash page, in a set of CPUs including said first CPU, where the address translation unit contents have been modified to reference a page other than said trash page, when said predetermined real address is received by the address translation unit of each CPU in said set.

15 05627979 May 6 1997 395/335 System and method for providing a
Jul 18 1994 graphical user interface for
mapping and accessing objects in
data stores

Inventor: Chang; Daniel T. et al.
Assignee: International Business Machines Corporation
Abstract:

A graphical user interface for mapping and accessing objects in data stores is disclosed. A user may define a mapping between object schema and data store schema by use of a high level language, Schema Mapping Definition Language (SMDL), which is data store independent, object oriented language independent, and extensible. The user may either write SMDL directly or generate SMDL through the use of a graphical user interface Smart Schema whose graphical semantics support the SMDL semantics. A Schema Mapping Internal Representation (SMIR) containing representations of the object schema, the data store schema, and the mapping of the object schema and the data store schema is generated by an

SMDL Parser from the SMDL. The SMIR is represented such that it may be accessible by both development interfaces and run-time environments. It supports the accessing of the mapping information given either the object schema or data store schema such that the data store schema may be accessed from the object schema, and the object schema may be accessed from the data store schema. An SMDL Generator may be used to generate the SMDL from the SMIR. The SMIR, SMDL Generator, SMDL Parser, and SMDL may be registered in a Data Store Manager (DSM) having a single, uniform, object oriented application programming interface for accessing one or more data stores, regardless of the type of data store. The DSM may use the SMIR to access objects from a data store. The SMIR may also be used by a run-time environment to provide direct access of objects from a data store, or it may be used by various Code Generators to generate an object oriented programming language for providing direct access to objects from a data store.

Exemplary Claim [2]:

2. A system for mapping and accessing objects in data stores, said system comprising:

graphical user interface for defining a mapping between an object schema and a data store schema in a high level language;

a first graphical user interface for mapping said object schema to said data store schema;

a data structure for embodying said mapping, said data structure supporting said first graphical user interface and supporting a run-time access;

high level language generator for generating, from said data structure, said high level language representing said mapping;

a parser for parsing said high level language into said data structure;

a translator for creating said first user interface representation of said mapping from said data structure;

a second user interface, dependent upon an application programming interface, for accessing object data from said data store;

said application programming interface, independent of said second user interface, for accessing object data from said data store;

a run-time component executing said application programming interface and implementing said object data access from said data store;

a data store access for said run-time component to utilize said data structure embodying said mapping definition in accessing said objects from said data store; and

a generator for generating, from said data structure, an object oriented programming language and data access language for deleting, adding, retrieving, and updating objects from said data store.

16 05550973 Aug 27 1996 395/182.08 System and method for failure
Mar 15 1995 recovery in a shared resource
system having a moving write lock

Status: certificate of correction has been issued

Inventor: Forman; Ira R. et al.

Assignee: International Business Machines Corporation

Abstract:

The system and method of the present invention provides for recovery from failure of a distributed processing system process designated as a master process for at least one shared resource. The system of the invention provides for detection of the failure by one or more of the shadow processes. The detecting process tests to determine whether it has the shared write lock managed by the master process. If it does, it becomes the master process. If not, it determines from the shared control

file which process holds the write lock and it communicates to that process a request to assume master process responsibilities.

Exemplary Claim [1]:

1. In a computer network that includes a plurality of processing nodes, a method for distributing control of a write lock for a shared data object from, a master processing node currently in control of the write lock, to one of the plurality of processing nodes requesting use of the write lock upon failure of the master processing node wherein the master processing node fails to transfer the write lock to the processing node requesting use of the write lock,

the method comprising the following steps in the sequence set forth:

detecting by the processing node requesting use of the write lock, when the master processing node fails to transfer the write lock to the processing node requesting use of the write lock;

determining, by the processing node requesting use of the write lock, whether the processing node requesting use of the write lock holds the write lock for the shared data object;

when the processing node requesting use of the write lock holds the write lock, requesting that the processing node requesting use of the write lock assume control over the write lock for that shared data object from the master processing node failing to transfer use of the write lock for that shared data object to establish a new master processing node for that shared data object;

when the processing node requesting use of the write lock does not hold the write lock for the shared data object, determining whether another processing node of the plurality of processing nodes holds the write lock for the shared data object;

when the another processing node holds the write lock for the shared data object, requesting that the another processing node assume control over the write lock for that shared data object from the master processing node failing to transfer use of the write lock for that shared data object to establish a new master processing node for that shared data object; and

when the another processing node fails to assume control of the write lock, executing a race among the plurality of processing nodes for control of the write lock for that shared data object from the master processing node failing to transfer use of the write lock for that shared data object to establish a new master processing node for that shared data object.

17 05448727 Sep 5 1995 395/612 Domain based partitioning and reclustering of relations in object-oriented relational database management systems
Apr 30 1991

Inventor: Annevelink; Jurgen

Assignee: Hewlett-Packard Company

Abstract:

A system and method of logically and physically clustering data (tuples) in a database. The database management system of the invention partitions (declusters) a set of relations into smaller so-called local relations and reclusters the local relations into constructs called domains. The domains are self-contained in that a domain contains the information for properly accessing and otherwise manipulating the data it contains. In other words, the data objects stored in the domains may be stored in a particular domain based upon a locality-of-reference algorithm in which a tuple of data is placed in a domain if and only if all objects referenced by the tuple are contained in the domain. On the other hand, the data objects stored in a domain may be clustered so that a tuple of data is placed in a domain based on the domain of the object referenced by a particular element of the tuple. By clustering the related object data

in this manner, the database management system may more efficiently cache data to a user application program requesting data related to a particular data object. The system may also more efficiently lock and check-in and check-out data from the database so as to improve concurrency. Moreover, versioning may be more readily supported by copying tuples of a particular domain into a new domain which can then be updated as desired.

Exemplary Claim [1]:

1. A relational database management system for storing and manipulating data objects having object identities for use as arguments to database functions, comprising:

means for defining and manipulating tuples of data representing domain based relationships of said data objects;

a relational database having storage tables therein for storing said tuples; and

storage management means for partitioning said storage tables, in accordance with domain based relationships of said data objects, into a plurality of local tables and for clustering said local tables into domains comprising collections of said local tables, each of said domains comprising a collection of local tables for storing a collection of data objects and the domain based relationships of said collection of data objects to other data objects in said relational database.

18 04525780 Jun 25 1985 395/490 Data processing system having a
May 31 1984 memory using object-based
information and a protection
scheme for determining access
rights to such information

Inventor: Bratt; Richard G. et al.

Assignee: Data General Corporation

Abstract:

A digital data processing system has a memory organized into objects containing at least operands and instructions. Each object is identified by a unique and permanent identifier code which identifies the data processing system and the object. The system uses a protection technique to prevent unauthorized access to objects by users who are identified by a subject number which identifies the user, a process of the system for executing a user's procedure, and the type of operation of the system to be performed by the user's procedure. An access control list for each object includes an access control list entry for each subject having access rights to the object and means for confirming that a particular active subject has access rights to a particular object before permitting access to the object. The system also includes stacks for containing information relating to the current state of execution of the system.

Exemplary Claim [1]:

1. In a digital computer system including

memory means for performing memory operations including storing and providing items of data, said items of data including instructions;

processor means connected to said memory means for performing operations in response to said instructions including processing said items of data;

memory organization means operative on said memory means for organizing said memory means into objects identified by unique identifiers and allowing the location of said items therein, each said item being associated with an object and addressable by a logical address specifying the unique identifier of the object with which said item is associated and an offset specifying the location of said item its associated object,

said memory organization means further identifying for each one of said objects a selected set of subjects representing entities for which

said processor means responds to said instructions;

memory operation specifier generation means in said processor means responsive to said instructions for providing a memory operation specifier for each item processed by said processor, said memory operation specifier for a given item specifying the object in which said given item is to be located and one of said memory operations; and

memory operations means responsive to a memory operation specifier which includes a logical address and a memory command specifying a memory operation for performing the memory operation specified by the memory command on the item specified by the logical address for a current subject for which said processor means is currently executing said instructions only when said current subject is one of the subjects in the selected set of subjects identified for the object specified in said memory operation specifier.

19 05335346 Aug 2 1994 395/609 Access control policies for an
Dec 12 1991 object oriented database, including access control lists which span across object boundaries

Inventor: Fabbio; Robert A.
Assignee: International Business Machines Corporation
Abstract:

The system and method of this invention provides an access control list which spans across object boundaries in an object oriented database. In addition to providing read and write access permissions, the access control list provides execute semantics which apply to the execution of methods in an object oriented database. Within the entries of the access control lists, each of the permissions for read, write, and execute can be assigned separately to each of a number of ids representing user ids or group ids. Upon request for access to the data by the user, the user id of the user and the group ids for which the user is a member are searched for within the entries to determine whether the user has the privileges to perform the operation requested against the objects. In addition, the access control policies are inherited from an object's superobject; resulting in a least privilege for the object.

Exemplary Claim [1]:

1. A method for controlling access privileges to data, said method comprising:

assigning at least one access control policy associated with a plurality of dynamically assignable groups across a plurality of dynamically extendable external objects in an object oriented database;

traversing objects to dynamically extend access control policies to encompass a newly extended object;

controlling a plurality of operations including an execute operation applied to execution of a plurality of methods to at least one of said plurality of dynamically extendable objects based on said assignment and at last one credential of a user requesting access to said data represented by at least one of said plurality of dynamically extendable; and

inheriting said assigned at least one access control policy by a second of said external objects descending from said at least one object; said inheriting further including determining a least amount of privilege associated with at least one composite object accessed by said user.

20 04498132 Feb 5 1985 395/490 Data processing system using
May 22 1981 object-based information and a protection scheme for determining

access rights to such information and using multilevel microcode techniques

Inventor: Ahlstrom; John K. et al.
Assignee: Data General Corporation
Abstract:

A digital data processing system has a memory organized into objects containing at least operands and instructions. Each object is identified by a unique and permanent identifier code which identifies the data processing system and the object. The system further uses multilevel microcode techniques for controlling sequences of microinstructions and for controlling the interval operations of the processor. The system uses a protection technique to prevent unauthorized access to objects by users who are identified by a subject number which identifies the user, a process of the system for executing a user's procedure, and the type of operation of the system to be performed by the user's procedure. An access control list for each object includes an access control list entry for each subject having access rights to the object and means for confirming that a particular active subject has access rights to a particular object before permitting access to the object. The system also includes stacks for containing information relating to the current state of execution of the system.

Exemplary Claim [34]:

34. In a digital computer system including processor means for performing operations on operands and memory means for storing instructions for controlling said processor means, said memory means being organized into objects for containing said operands and instructions, said processor means comprising:

means for uniquely identifying said objects, including

memory organizing means responsive to first instructions for designating locations in said memory means as objects for containing items of information, and

means for generating unique identifier codes, a unique idendifier code being uniquely and permanently associated with a corresponding object generated by said processor means,

means for addressing said operands comprising:

name table means for storing name table entries, each name table entry corresponding to a name of a procedure and each name table entry comprising data resolvable to provide a location in said memory means of an operand referred to by said corresponding name, and

means responsive to said names for resolving each name table entry so as to provide outputs to said memory means representing locations in said memory means of said operands,

protection means for preventing a user, currently using said digital computer system to execute a program comprising a procedure object containing selected instructions, from obtaining unauthorized access to selected objects, including:

subject number memory means responsive to the operation of said processor means for storing a currently active subject number of a plurality of subject numbers,

each subject number corresponding to one of a plurality of subjects wherein each subject comprises the combination of (1) a user of said digital computer system, (2) a process of said digital computer system for executing a procedure of said user, and (3) the type of operation to be performed by said digital computer system in response to an instruction of a procedure of said user, and

said currently active subject number identifies the user currently utilizing said digital computer system, (2) a process currently executing

a procedure of said user's program, and (3) the type of operation to be performed by said digital computer system in response to a a current instruction of a procedure of said user's program,

protection memory means for storing at least one access control list, each access control list corresponding to an object and comprising at least one access control entry, each access control entry corresponding to a subject having access rights to said corresponding object and containing the access rights of said subject to said corresponding object, and

access rights means responsive to a currently active subject number and to the operation of said processor means for indexing said protection means in response to a current instruction of a procedure of said first user's program when said current instruction requests access to an object and for comparing the access rights of a currently active subject to said object,

ALU means for performing operations directed by said instructions.

first microcode means for storing first selected sequences of microinstructions for controlling operations of said ALU means directed by said instructions, said instructions being S-Language instructions having a uniform, fixed format,

said first microcode means responsive to said instructions to provide corresponding first selected sequences of microinstructions to said ALU means, and

second microcode means connected to said bus means for storing second selected sequences of microinstructions for controlling internal operations of said processor means,

said second microcode means responsive to the operation of said processor means to provide said second selected microinstruction sequences to said ALU means.

21 05412806 May 2 1995 395/602 Calibration of logical cost
Aug 20 1992 formulae for queries in a heterogeneous DBMS using synthetic database

Inventor: Du; Weimin et al.
Assignee: Hewlett-Packard Company
Abstract:

A programmable machine system and method for managing electronic data access among multiple different relational databases in a network distributed database environment. The machine is programmed so that it can construct cost-effective access strategies for any of the participating databases absent any DBMS-specific cost models. The system provides query optimization across different database management systems in a network distributed database environment based on a calibrating database relying only on typical relational database statistics and cost data is developed by running queries in the various databases against the calibrating database. A logical cost model is constructed using the resulting cost data and is used to estimate the cost of a given query based on logical characteristics of the DBMS, the relations, and the query itself. The cost of a complex query is estimated using primitive queries. Optimal query access strategies are thereby designed and used to control execution of the queries across relational databases controlled by two or more different database management systems.

Exemplary Claim [1]:

1. A database access system for optimizing database queries in a heterogeneous distributed database system, the system comprising:

a first database machine incorporating a first relational database management system and accompanying first database;

a second database machine incorporating a relational database

management system and accompanying second database;

the first and second relational database management systems being different but conforming at least to a predetermined structured query language (SQL);

communication means for electronic bidirectional communications between the different database machines;

means coupled to the communication means for sending and receiving an electronic message to and from any of the database machines, the message containing data defining a database query;

a data access logical cost model comprising logical cost formulae for optimizing queries in each database in the system;

a synthetic database for use in calibrating the data access logical cost model for each relational database management system in the distributed database system;

means for querying the synthetic data base on each database machine to determine cost coefficients for use in said logical costs formula to calibrate the data access logical cost model: and

means responsive to a database query for accessing each of the first and second databases of said first and second database machines in accordance with a least cost index obtained from said data access logical cost model.

22 05463733 Oct 31 1995 395/182.08 Failure recovery apparatus and
Aug 8 1994 method for distributed processing
shared resource control

Inventor: Forman; Ira R. et al.
Assignee: International Business Machines Corporation
Abstract:

Communicating the failure of a master process controlling one or more shared resources to all process sharing the resources. A shared resource control file is established that contains the identities of all sharing processes. Master process failure triggers a race to establish exclusive access over the shared control file. The new master reads shadow address data from the old shared control file, marks it as invalid and establishes a new control file based on renewed registrations from the sharing processes. The master process maintains the sharing process list as processes begin and end sharing.

Exemplary Claim [1]:

1. A method for managing recovery of a distributed processing system in which shared resources are each controlled by a master process, the distributed processing system having a plurality of processors each executing a plurality of processes and each controlled by a separate operating system, each of said processors having memory, and each of said processors interconnected to the other processors by means of a communications network, the method comprising the steps of:

detecting failure of a master process for a shared resource;

requesting exclusive access to a control file using a network file system management procedure independent of said operating system for said processors, if said detecting step detects a failure;

establishing exclusive access to said control file using said network file system management, if said exclusive access request is granted;

determining from said control file all other processes accessing said shared resource;

invalidating said control file;

sending a message to each of said other processes indicating failure of said master process; and

creating a new control file and entering data for each process responding to said message.

23 05303367 Apr 12 1994 395/613 Computer driven systems and
Dec 4 1990 methods for managing data which
use two generic data elements and
a single ordered file

Inventor: Leenstra, Sr.; Richard B. et al.
Assignee: Applied Technical Systems, Inc.
Abstract:

Computer-based systems and methods for managing data. These systems and methods take advantage of a unique model which: increases speed and flexibility; eliminates the need for a complex data manipulation language, data or application dependent software, and separate structuring tools such as pointers, lists, and indexes; and automatically creates among the data relationships which may or may not have been apparent to a user or the designer. Salient, unique features of the systems and methods are their capabilities for providing: a generic data structure consisting of two generic data elements and a generic data set array, indentured data set relationships, inversion of data set relationships, dynamic reorganization of data sets, control over the data relationships that can be established, global linking of data sets, and automatic connection projection. A database structure which is independent of the type of data being managed is established, and the data is entered into a single file in which the data is automatically logically and physically ordered.

Exemplary Claim [1]:

1. A method of employing a computer with input means, memory, and a data processing unit to manage data, said method comprising the steps of:

employing said computer to create an original array of data sets, each said data set having a data type element for differentiating one kind of subject matter from another and a data value element for differentiating data sets with the same data type element and said array containing a key data set and data sets related to said key data set;

employing said computer to link the related data sets in a hierarchy in which the related data sets are each in an indentured relationship to the key data set;

so employing the computer to invert the relationship between an indentured data set in the original array and the remaining data sets in said array as to create an inverted data set array having: (a) the data set entered in the original array as its key data set, and (b) all data sets higher in the hierarchy of the original data set linked to that key data set in inverted order; and

storing the original and inverted data set arrays in the memory of the computer.

24 04995112 Feb 19 1991 380/25 Security system
Jun 30 1989

Inventor: Aoyama; Mitsunobu
Assignee: Kabushiki Kaisha Toshiba
Abstract:

In a distributed processing system having a plurality of nodes, a correspondence table, which represents a correspondence between the nodes and corresponding passwords that grant access to the nodes, stored in a disk device. Each of the nodes includes a passthrough device which permits passthrough of access request data to a node of the next hierarchical level. Upon receipt of access request data, the passthrough device refers to the correspondence table to check a password contained in the received access request data for validity of access to the node. When the password is valid, the passthrough of the access request data is permitted.

Exemplary Claim [1]:

1. A security system for use in a distributed processing system having

a plurality of nodes, comprising:

passthrough means for passing access request data through an intra node to another one of said plurality of nodes, said access request data including at least address information for specifying one of said plurality of nodes and security information for obtaining access to one of said plurality of nodes;

node directory storing means for storing a node directory representing a correspondence between each of said plurality of nodes and respective security information; and

control means for receiving said access request data, and upon determining that said address information does not specify said intra node and that said security information of said access request data corresponds to said address information in accordance with said correspondence represented by said node directory, for causing said passthrough means to pass said access request data through said intra node to said other one of said plurality of nodes.

25 05526524 Jun 11 1996 395/726 Method and system for management
Dec 23 1993 of locked objects in a computer supported cooperative work environment

Status: certificate of correction has been issued
Inventor: Madduri; Hari H.
Assignee: International Business Machines Corporation
Abstract:

A method and system for efficiently managing access to multiple objects which are stored within a distributed data processing system wherein access to those objects may be selectively locked by one of the users enrolled within the distributed data processing system. Each time access to an object within the distributed data processing system is attempted, the lock status of that object is determined. If access to the object is not locked, access is granted to the requesting user. However, if access to the object is currently locked, the identity of each requesting user is stored within a "camp-on" table within the distributed data processing system. Next, the lock status of the object is determined. This may be accomplished by detecting the release of an object by a user who has previously locked that object or by periodically checking the lock status of all objects. When the lock status changes, an audible or visual notification is automatically transmitted to each user who has requested access to that object, greatly enhancing the efficiency of access management to objects stored within a distributed data processing system.

Exemplary Claim [1]:

1. A method in a distributed data processing system having a plurality of objects stored therein to which access may be selectively locked by one of a plurality of users enrolled within said distributed data processing system, for efficiently managing access to said plurality of objects, said method comprising the data processing system implemented steps of:

determining a lock status for an object stored within said distributed data processing system in response to an attempted access of said object by a selected user within said distributed data processing system;

granting access to said object by said selected user in the event access to said selected object is not locked; and

in the event access to said object is locked:

determining if said selected user has requested a "camp-on" for said object;

notifying said selected user that said object is locked if said selected user has not requested a "camp-on" for said selected object;

if said selected user has requested a "camp-on" for said selected

object:

storing an identity of said selected user within said distributed data processing system;

periodically determining a lock status of said object; and

automatically notifying said selected user within said distributed data processing system in response to a change in lock status of said object by means of a human perceptible indication at a location of said selected user.

26 05551027 Aug 27 1996 Sep 11 1995 395/617 Multi-tiered indexing method for partitioned data

Inventor: Choy; David M. et al.

Assignee: International Business Machines Corporation

Abstract:

A multi-tiered indexing method is disclosed for a partitioned table in a parallel or distributed database system. A Local Index is created and maintained for each partition of the table and a Coarse Global Index is created and maintained. The Coarse Global Index identifies the indexed partition(s) by partition identifiers (PIDs) and associates the individual Index Key Values with their target partitions so that an access request with a highly partition-selective search predicate on the Index Key can be quickly and easily directed to the target partition(s) for processing. An index maintenance locking protocol is also disclosed which handles the insertion and deletion of index entries and assures the consistency between the Local Index entries and the Coarse Global Index entries during concurrent index accesses by different transactions. The locking protocol minimizes locking only to those cases involving an inserted or deleted key and to the key following and possibly the key preceding the inserted or deleted key to allow high concurrency between simultaneous Readers, Inserters, and Deleters. This method enhances the efficiency of complex query evaluation and index maintenance and attains a high throughput for transaction processing.

Exemplary Claim [1]:

1. A method for maintaining consistency between a second index table and a first index table in a multi-tiered index structure to handle the insertion of an index entry into a unique index, the multi-tiered index structure including a respective second index table of index key values corresponding with each respective partition of a partitioned database and a first index table containing at least one unique first index entry for each index key value in each of the respective second index tables,

i) wherein the second index table has a second index identifier and contains second index entries, each of the second index entries having an identifier which identifies an object in the respective database partition corresponding with the second index table, and having a second index key value which relates to the identified object, at least some of the index key values being distinct, and

ii) wherein the first index table has at least one first index entry for each distinct one of the second index key values in each second index table, each of the first index entries having a second index identifier which identifies a second index table, and having a first index key value which relates to one of the second index key values in the identified second index table,

the method comprising the computer-implemented steps of:

A) determining if the second index key value in the index entry to be inserted is already present in the second index table;

B) if the second index key value in the index entry to be inserted is already present in the second index table, then rejecting the insertion; and

C) if the second index key value in the index entry to be inserted is not already present in the second index table, then performing the steps of:

1) inserting the index entry into the second index table;

2) determining if the first index key value which relates to the second index key value is already present in the first index table, and

3) if the first index key value which relates to the second index key value is already present, performing the steps of:

i) deleting the inserted second index entry from the second index table; and

ii) rejecting the insertion;

4) determining if the first index key value which relates to the second index key value is present in the first index table, and,

5) if the first index key value which relates to the second index key value is not present, then inserting, into the first index table,

i) a first index entry having a first index key value relating to the second index key value; and

ii) a second index identifier identifying the second index table.

27 05241673 Aug 31 1993 395/614 System for garbage collecting unused memory space represented by a digraph by assigning values of node identifiers to selected variables based upon predetermined conditions
Oct 1 1990

Inventor: Schelvis; Marcellinus A. J.
Assignee: Oce-Nederland B.V.
Abstract:

A method of distributing status information is described, regarding a digraph in a logically organized system of groups of entities related to each other and to be represented by digraphs, and a device for using such a method. In a system which comprises a number of groups which may be represented by digraphs, the lacking of a node having a particular status in a group is detected by the method. When applying the method to a distributed object-oriented system, in which a root is such a node having a particular status, incremental distributed garbage collection is realized. The invention is also effective in the case in which cycles or subcycles occur in graph of objects referring to each other. Upon realizing garbage collection, inaccessible data information occupying memory space is removed from memory to thereby free previously occupied memory space.

Exemplary Claim [1]:

1. A method of garbage collecting inaccessible information to reclaim memory space previously occupied in a memory of a logically organized computer network system including an assembly of groups of related entities, each of the groups being represented by a digraph without subcycles and each of the entities being represented by a node of the digraph, by distributing status information regarding the digraph over each of the nodes of the digraph, in which an identifier of a unique value is associated with each node, per digraph, and in which a set of elements is associated with each node not having a particular status, each element representing a variable which is coupled separately to an immediate predecessor of the associated node, the method comprising the steps of:

(a) assigning the value of the identifier of a node having a particular status to each variable coupled to the node;

(b) assigning the value of the identifier of a node without a particular status to each variable which is coupled to the node if the identifier satisfies a predetermined comparison criterion with respect to

the variables of the set associated with the node;

(c) assigning the value of a variable of the set associated with a node to each variable which is coupled to the node, if all the variables of the set associated with the node are identical and the identifier of the node fails to satisfy the predetermined comparison criterion of step (b); and

(d) removing each variable which is coupled to the node if, in the set associated with the node, each variable is equal to the value of the identifier of the node or the set associated with the node is empty to thereby remove information from memory which has been determined to be inaccessible.

28 05263157 Nov 16 1993 395/200.59 Method and system for providing
Feb 15 1990 user access control within a distributed data processing system by the exchange of access control profiles

Inventor: Janis; Frederick L.
Assignee: International Business Machines Corporation
Abstract:

A method is disclosed for providing user access control for a plurality of resource objects within a distributed data processing system having a plurality of resource managers. A reference monitor service is established and a plurality of access control profiles are stored therein. Thereafter, selected access control profiles are exchanged between the reference monitor service and a resource manager in response to an attempted access of a particular resource object controlled by that resource manager. The resource manager may then control access to the resource object by utilizing the exchanged access control profile. In a preferred embodiment of the present invention, each access control profile may include access control information relating to a selected user; a selected resource object; a selected group of users; a selected set of resource objects; or, a predetermined set of resource objects and a selected group of users.

Exemplary Claim [1]:

1. A computer implemented method of providing user access control for a plurality of resource objects within a distributed data processing system having at least one reference monitor service and a plurality of resource managers associated with said plurality of resource objects, each of said plurality of resource managers controlling access to different selected ones of said resource objects, each of said resource managers associated with a reference monitor service, said method comprising the computer implemented steps of:

storing a plurality of unique access control profiles within each said reference monitor service, wherein selected ones of said plurality of access control profiles each include access control information relating to a predetermined set of said resource objects and a selected list of users each authorized to access at least a portion of said predetermined set of resource objects;

querying an associated reference monitor service by a selected one of said resource managers in response to an attempted access of a particular resource object among said plurality of resource objects, wherein access to said particular resource object is controlled by said selected resource manager;

transmitting a selected access control profile associated with said particular resource object from said associated reference monitor service to said selected one of said resource managers if said selected access control profile existed in said associated reference monitor service; if

not, attempting to retrieve said selected access control profile from another said reference monitor service and thereafter transmitting said retrieved access control profile to said selected one of said resource managers;

utilizing said selected resource manager to control access to said particular resource object in accordance with access control information in said selected access control profile; and

denying access to said particular resource object in response to a failure to retrieve said selected access control profile.

29 05295262 Mar 15 1994 395/608 Read-only access without blocking
May 16 1991 via access vectors

Inventor: Seigh, II; Joseph W.

Assignee: International Business Machines Corporation

Abstract:

A system and method for maintaining linked data structures stored in a computer system capable of processing the stored data as addressable object nodes, such that any data object node may be modified prior to the completion of outstanding read only accesses to that node. The system and method maintain an access vector for each node. The access vectors include an access counter which counts the number of read only accesses from a present node to the next node, and a link pointing to a next node. The number of read only accesses in effect for a node is the sum of access counts of all access vectors pointing to that node, minus the ADJ and minus the sum of access counts of all access vectors with access to that node. A node in the list can be replaced or deleted by first changing the pointers of all nodes that point to the node to point to a new one. Then, modifying the access count of the node by subtracting the maximum value of the access counts pointing to the node from the count in the node. Read only accesses out of the node continue to increment the access count, and when it reaches zero the node can be deleted. Because the pointers to the node had been changed, read only processes can continue without pausing for the change or replacement event.

Exemplary Claim [1]:

1. A computer implemented method for maintaining a linked data structure stored in a computer system capable of processing the stored data as addressable object nodes, such hat any node may be modified prior to the completion of outstanding read only accesses to said any node, comprising the steps of:

maintaining an access vector for each of the nodes, each access vector including an access counter for maintaining an access count value corresponding to how many processes have read access to a first node, and a link for maintaining a pointer to a next node;

adding one or more links to the data structure, wherein when a link associated with an access vector is added to point to a second node, the value of the access count of the second node and the value of the access counts of all nodes indirectly pointed to by the added link are incremented by the value of the access count of the access vector associated with the added link;

deleting links from the data structure, wherein when a link associated with an access vector is deleted from a third node, the value of the access count of the third node and the value of the access counts of all nodes previously indirectly pointed to by the deleted link are decreased by the value of the access count of the access vector associated with the deleted link; and

deleting nodes by:

deleting all links to a fourth node,

waiting until an access count for said fourth node becomes zero, said

access count becoming zero as readers complete their respective read operations at said fourth node,

deleting all links from said fourth node to further nodes, and

deallocating said fourth node.

30 04991079 Feb 5 1991 395/200.31 Real-time data processing system
Aug 30 1989

Inventor: Dann; James C.

Assignee: Encore Computer Corporation

Abstract:

A real time data processing system in which each of a series of processing nodes is provided with its own data store partitioned into a first section reserved for the storage of data local to the respective node and a second section reserved for the storage of data to be shared between nodes. The nodes are interconnected by a data link and whenever a node writes to an address in the second section of a data store the written data is communicated to all of the nodes via the data link. The data in each address of the second sections of the data stores can be changed only by one respective processing node which acts as a master for that address. As each address containing shared data can only be written to by one node collisions between different nodes attempting to change a common item of data cannot occur.

Exemplary Claim [1]:

1. A real-time data processing system, comprising at least two processing nodes, a local data store in respect of each processing node, each local data store being partitioned into sections a first one of which is reserved for the storage of non-shared data local to the respective processing node and a second one of which is reserved for the storage of data to be shared between processing nodes, a data link interconnecting the processing nodes, generating means at each processing node for generating a read/write message including an address and data to be written to that address whenever that processing node writes to an address in the second section of the local data store, transmitting means for transmitting each generated write message via the data link to each of the processing nodes, allocating means for allocating to each address in the second sections of the local data stores a respective processing node which is to be a master processing node for that address, preventing means for preventing data being written to any address in the second section of a local data store other than by the allocated master processing node, said allocating an preventing means including an address range comparator in respect of each processing node for comparing the address of a data write message generated by that processing node with a preset range of addresses and for transferring the data write message to the data link only if the compared address is within the preset range, each processing node including a latch into which each data write message is loaded, a first in first out register connected to the output of the latch, a detector for detecting a successful writing of data to the data store local to the processing node, an AND gate connected to the detector and address range comparator and controlling the latch such that the content of the latch is transferred to the register when the compared address is within range and successful writing of data is detected, and a transmitter connected to the register for transmitting messages stored in the register over the data link.

31 05072373 Dec 10 1991 395/200.31 Real-time data processing system
Jan 17 1991

Inventor: Dann; James C.

Assignee: Encore Computer U.S., Inc.

Abstract:

A real time data processing system in which each of a series of processing nodes is provided with its own data store partitioned into a first section reserved for the storage of data local to the respective node and a second section reserved for the storage of data to be shared between nodes. The nodes are interconnected by a data link and whenever a node writes to an address in the second section of a data store the written data is communicated to all of the nodes via the data link. The data in each address of the second sections of the data stores can be changed only by one respective processing node which acts as a master for that address. As each address containing shared data can only be written to by one node collisions between different nodes attempting to change a common item of data cannot occur.

Exemplary Claim [1]:

1. A method of reflecting data among a plurality of processing nodes interconnected through a reflective data link, comprising the steps of:

establishing for each node a memory having a first data port and a second data port and having a set of addresses;

establishing for each node processing means connected to said first data port of the associated memory for reading data from said memory and for writing data into said memory;

sensing write data from the processing means of a node to the first data port of the associated memory that is being written into preselected addresses of said associated memory, and

forwarding said sensed write data directly into said data link for reflection into the memories of other processing nodes interconnected through the data link,

sensing write data on said data link being written into preselected addresses of memory, and

forwarding said sensed write data from said data link directly into the associated memory of a node through the second data port of the memory.

32 05581732 Dec 3 1996 395/475 Multiprocessor system with
Oct 13 1993 reflective memory data transfer device

Inventor: Dann; James C.

Assignee: Encore Computer, U.S., Inc.

Abstract:

A real time data processing system in which each of a series of processing nodes is provided with its own data store partitioned into a first section reserved for the storage of data local to the respective node and a second section reserved for the storage of data to be shared between nodes. The nodes are interconnected by a data link and whenever a node writes to an address in the second section of a data store the written data is communicated to all of the nodes via the data link. The data in each address of the second sections of the data stores can be changed only by one respective processing node which acts as a master for that address. As each address containing shared data can only be written to by one node collisions between different nodes attempting to change a common item of data cannot occur.

Exemplary Claim [1]:

1. A processing system comprising:

plural processing nodes, each node comprising a node memory having a portion of which constitutes a shared store, and a data write means for writing data to the node memory;

data reflecting means for reflecting data written by a first data write means of a first node to the shared store of said first node and

writing said data to other shared stores of other nodes independently of the data write means of all said nodes;

a memory mapped input/output; and

a random access memory coupled to said data reflecting means and said memory mapped input/output and storing the memory mapped input/output data.

33 05613079 Mar 18 1997 395/468 System for verifying the proper
Sep 5 1995 operation of a replication
facility

Inventor: Debique; Kirt et al.
Assignee: Microsoft Corporation
Abstract:

A verification strategy is provided to verify proper multi-master replication of logical structures, such as objects, in a data processing system. The strategy is especially well adapted for use within the distributed environment. The strategy verifies that proper reconciliation of name spaces has occurred via multi-master replication. The strategy also verifies that the correct propagation of knowledge of changes to objects has occurred during replication.

Exemplary Claim [1]:

1. In a computer system having a storage with a name space containing logical structures and a multi-master replication facility for replicating said logical structures, a method for verifying proper operation of said replication facility, comprising the computer-implemented steps of:

providing a testing tool for verifying proper operation of said replication facility;

the testing tool making changes to selected ones of said logical structures within the name space at a first location in said computer system;

the testing tool logging information about said changes in a log held in said storage;

the replication facility replicating said selected logical structures to a second location in said computer system; and

the testing tool verifying that said changes to said selected logical structures were properly replicated to said second location in said replicating step to verify proper operation of said replication facility by examining the logged information in the log.

34 04731734 Mar 15 1988 395/412 Digital computer system
Feb 14 1986 incorporating object-based
addressing and access control and
tables defining derivation of
addresses of data from operands in
instructions

Inventor: Gruner; Ronald H. et al.
Assignee: Data General Corporation
Abstract:

A digital computer system having a memory system organized into objects for storing data and a processor for processing data in response to instructions. An object identifier and an access control list are associated with each object. The memory system responds to logical addresses for data which specify the object containing the data and the offset of the data in the object and to a current subject for which the processor is referencing the data. The memory system performs a memory operation for the processor only if the access control list for the object specified by the logical address allows the current subject to perform the desired memory operation. The objects include procedure objects and data

objects. The procedure objects contain procedures including the instructions and name tables associated with the procedures. The instructions contain operations codes and names representing data. Each name corresponds to a name table entry in the name table associated with the procedure. The name table for a name contains information from which the processor may derive the logical address for the data represented by the name. The processor may then use the logical address to specify a memory operation on the data represented by the name.

Exemplary Claim [1]:

1. A digital computer system comprising:

memory means for storing and providing data items, said data items including instructions,

memory organization means operative on said memory means for organizing said memory means into objects which provide for the location of said data items in said memory means, each object being identifiable by an object identifier;

access control means for identifying for each object a set of subjects which are permitted to access the data items in said object and for identifying for each subject a specified set of memory operations which each said subject is permitting to perform;

means responsive to said access control means and to a request from a current subject for access to a current data item in an object and for the performance of a current memory operation for determining whether said current subject is a subject which is permitted to access said current data item and whether said current memory operation is one which said current subject is permitted to perform;

memory oepration means responsive to a memory operation specifier which includes

a logical address specifying an object identifier and a location in the object identified by said object identifier, and

a memory command specifying a current memory operation,

and further responsive to a request from a current subject for performing a current memory operation specified by said memory command when said determining means determines that said current subject is one having permission to access the object specified by said logical address and to perform said current memory operation,

and further wherein said instructions include

operation codes specifying operations, including memory operations, of said digital computer system, certain instructions further including

a name representing a data item to be used in an operation specified by an operation code, and

said system further includes

means for storing a plurality of name table entries each name table entry corresponding to a data item and to the name representing said data item and each name table entry including information from which the logical address of the data item represented by the name corresponding to said name table entry can be derived, and

processor means connected to said memory means and including

means for providing instructions from said memory means,

instruction decoding means responsive to instructions from said instruction providing means for decoding a current instruction to provide one or more names therein,

logical address generation means responsive to the information in a name table entry corresponding to a name for deriving a logical address from said information, said logical address generating means including

name resolution means responsive to a name in said decoded instruction and to the information in the name table entry corresponding to said name for generating the logical address for the data item represented by said

name, and

next instruction address generation means further responsive to said decoded instruction for providing a logical address of a next current instruction, and

control means responsive to a name from said instruction decoding means and to the logical address from said logical address generation means for providing a representation of a current subject and one or more memory operation specifiers to said memory operation means.

35 05555375 Sep 10 1996 395/200.56 Method and apparatus for network
Jan 12 1995 computer systems management group
administration

Inventor: Sudama; Ram et al.
Assignee: Digital Equipment Corporation
Abstract:

The following is a method and apparatus for administering an operation specified for performance on a set of independently managed hosts. The operation is received initially by one of a plurality of management servers in a managed host system. The operation, specifying a group object, is transferred to the management server designated by the system to administer the operation specified on the group object. The designated management server thereafter decomposes the group object into constituent objects which may be host objects or additional group objects. Next, the locally administered objects are scheduled for execution on the hosts administered by the designated management server. The non-locally administered objects are transferred to the management servers identified in a database for administering the objects. After executing the operation on the host objects, the host objects and group objects return status information back to the designated management server. The status information is then transmitted to the management server that initially received the operation.

Exemplary Claim [1]:

1. A method for administering an operation on a network of computer systems, said network of computer systems having a plurality of management servers for administering objects, a first group object having a first plurality of constituent objects and a second group object having a second plurality of constituent objects, said method comprising the steps of:

creating a name space containing a list of all objects including said first group object, said second group object, each of said first plurality of constituent objects and each of said second plurality of constituent objects, said list identifying said first plurality of constituent objects as belonging to said first group object and identifying said second plurality of group objects as belonging to said second group object;

identifying in said list an associated one of said plurality of management servers for administering each of said first group object, said second group object, each of said first plurality of constituent objects and each of said second plurality of constituent objects;

generating an operation that is directed to a first target object selected from the group consisting of said first group object and said second group object, said operation comprising at least one sub-operation directed to a second target object selected from the group consisting of said first group object, said second group object, each of said first plurality of constituent objects and each of said second plurality of constituent objects, said second target object being different from said first target object;

receiving said operation at the management server identified in said list as administering said first target object;

transferring said operation to said first target object for execution;

transferring said sub-operation to the management server identified in said list as administering said second target object;

transferring said sub-operation to said second target object for execution.

36 05448726 Sep 5 1995 395/614 Data base management system with
Oct 23 1989 data dictionary cache including a
single loadable object descriptor

Inventor: Crumsie; William J. et al.
Assignee: International Business Machines Corporation
Abstract:

In a data base system management system having a dictionary for maintaining a list of sets of data associated with each of a plurality of user application programs, a method and apparatus for decreasing the data base access time by forming a data model of the logical relationship of said list, generating static access modules, creating a secondary data model in a pre-built machine executable format and storing the data model. In the execution stage accessing the secondary data model and manipulating data values using the secondary data model.

Exemplary Claim [1]:

1. In a computer system in which one or more data files containing specific instances of data of interest to a user application are accessed using a data model defining said files to the application, said computer system having a main memory and a permanent storage subsystem, a method of accessing said one or more data files comprising the steps of:

generating a data model defining said one or more data files to said application as a single loadable object transferable from said storage subsystem to said main memory in a single data access to said storage subsystem;

caching said data model as a single loadable object in said main memory; and

using said data model cached in said main memory to access said one or more data files from said application.

37 05255369 Oct 19 1993 395/200.44 Multiprocessor system with
Sep 11 1991 reflective memory data transfer
device

Inventor: Dann; James C.
Assignee: Encore Computer U.S., Inc.
Abstract:

A real time data processing system in which each of a series of processing nodes is provided with its own data store partitioned into a first section reserved for the storage of data local to the respective node and a second section reserved for the storage of data to be shared between nodes. The nodes are interconnected by a data link and whenever a node writes to an address in the second section of a data store the written data is communicated to all of the nodes via the data link. The data in each address of the second sections of the data stores can be changed only by one respective processing node which acts as a master for that address. As each address containing shared data can only be written to by one node collisions between different nodes attempting to change a common item of data cannot occur.

Exemplary Claim [1]:

1. A data processing system comprising a plurality of nodes and a data bus interconnected therewith,

at least one processing node comprising:

a dual port local data store partitioned into sections, a first section reserved for the storage of data local to the node and not to be

shared with the other nodes and a second section reserved for data shared with other processing nodes;

a processor;

a local bus connecting a first port of the local data store to the processor for transferring data to the local data store; and

read/write sense logic means connected to the local bus for sensing when the processor is transferring data to the local store, connected to the data link for transmitting sensed data onto the data link for sharing with another node and for sensing received data from the data link which is being shared from another node and for inputting sensed received data through a second port of the local data store; said read/write sense logic means comprises:

first address comparator means for comparing an address of a signal transmitted by the processor to said first port, to determine if the signal is within a first range of allowable addresses for shared data;

successful write detector means for detecting the presence of a successful write signal on the local bus;

first latch means for holding data transmitted by the processor on the local bus;

first gate means for receiving signals from the first address comparator means and the successful write detector means and for activating the first latch means upon receipt of signals indicating the address is within the first range of allowable addresses and a successful write to memory has occurred;

first FIFO register means for receiving data from the first latch means responsive to activation of the first gate means;

bus request logic means for generating a bus request signal to transmit data stored in the first FIFO register means along the data bus and for releasing data from the first FIFO register means upon receipt of a bus grant signal from another node designated as master node;

transmitter means for receiving data from the first FIFO register means and transmitting the data along the data bus to other nodes responsive to receipt of the bus grant signal by the bus request logic means;

receiver means for receiving signals from the data bus;

data valid detector means for detecting the presence of a data valid signal on the data bus;

second address comparator means for receiving an address of a signal received by the receiver means and determining if the address is within a second range of allowable addresses for shared data;

second latch means for holding data signals received by the receiver means;

second gate means for receiving signals from the data valid detector means and the second address comparator means and for activating the second latch means upon receipt of signals indicating the address is within the second range of allowable addresses and a valid data signal was received by the data valid detector means;

second FIFO register means for receiving data from the second latch means responsive to activation of the second latch means;

memory transfer request means for generating a signal requesting access to the local data store through the second port of the local data store and for releasing data from the second FIFO register means to the local data store upon receipt of a signal granting access to the local data store.

38 04774661 Sep 27 1988 395/603 Database management system with
Nov 19 1985 active data dictionary
Inventor: Kumpati; Murari

Assignee: American Telephone and Telegraph Company, AT&T Information Systems
Abstract:

This invention pertains to database management systems and, in particular, to a database management system which has an active data dictionary that the user can both access and modify. The user makes use of simple commands to control, order and query not only the underlying data controlled by the database management system but also the contents of the data dictionary. This capability enables the user to write generic application programs which are logically independent of the data since the subject database management system enables the user/application program to access all data in the database independent of each application program's data model.

Exemplary Claim [1]:

1. A database management system which runs on a computer serving a plurality of users and which interfaces a plurality of user application programs to a database for managing sets of data stored on said database comprising:

data dictionary means for specifying relationships between each of said user application programs and said sets of said data stored in said database associated with each of said user application program;

data model means for controlling the definition of the specified relationships of said data dictionary;

query means responsive to change commands from a user application program for modifying said data model;

wherein said query means includes means responsive to said change commands for changing said data dictionary to correspond to said data model changes; and

said query means further includes means responsive to a data request from said user application program for accessing the requested sets of said data from said database.

39 05544353 Aug 6 1996 395/608 Distributed processing object
Jun 14 1993 shared resource control apparatus
and method

Inventor: Forman; Ira R. et al.
Assignee: International Business Machines Corporation
Abstract:

A system and method for determining a master process for control of a shared system resource. The improved system requires the master process to hold exclusive access on a shared resource control file only intermittently. The master process periodically updates the shared resource control file with a new timestamp. Processes seeking resource access read the shared control file and determine whether another process has been designated master. If the interval since the latest timestamp is greater than a preset staleness interval, the shared control file is discarded and a new one created by the accessing process.

Exemplary Claim [1]:

1. A method of determining a master process for control of a shared resource in a computer system having a plurality of processes operating on at least one processor that has a memory and access to a shared data storage means, the method comprising:

testing said shared data storage means for the presence of a shared resource control file for said shared resource;

if no file exists, creating a shared resource control file in said shared data storage means, writing master process identification information to said shared resource control file, and writing a timestamp to said control file;

if a file exists, requesting exclusive access to said file;

if access denied, waiting and retrying until exclusive access to said file is obtained;

if access granted, determining the difference between current time and the last time stamp;

if said difference is less then a first preset interval, designating the requesting process as a shadow process;

if said difference is greater than said first preset interval, discarding said shared resource control file, creating a new shared resource control file and writing master process identification information to said shared resource control file, and writing a timestamp to said control file;

if said requesting process is a master process for said shared resource, replacing the timestamp in said shared resource control file with a current timestamp after a preset second interval has passed.

40 05313629 May 17 1994 395/614 Unit of work for preserving data
Oct 23 1989 integrity of a data-base by creating in memory a copy of all objects which are to be processed together

Status: certificate of correction has been issued

Inventor: Abraham; Robert L. et al.

Assignee: International Business Machines Corporation

Abstract:

A Unit of Work object class for an object oriented database management system provides concurrent processing through Unit of Work levels and instances while maintaining the integrity of the data in the database. Each new Unit of Work assigned to a task is an instance of the Unit of Work object class. A Unit of Work manager controls each step such that manipulation of the data occurs to the copies at that particular level for that particular instance. Only after all levels have been completed satisfactorily will a "Commit" occur to the data in the database. If completion is not satisfactory, Rollback of the levels occur, thus preserving data integrity. The Unit of Work manager can also switch control between Unit of Work instances, thus permitting simultaneous performance of tasks.

Exemplary Claim [1]:

1. An object management system for an object oriented computing system executing on a data processing system, said object oriented computing system including a plurality of objects which are stored in nonvolatile storage and which are processed in volatile storage, each object including an object frame containing data attributes, and at least one object method for performing actions on the associated object, said object management system comprising:

a unit of work object class including a unit of work object frame containing pointers to objects which are processed together by said object oriented computing system, and including a first method for loading said objects which are processed together from nonvolatile storage to volatile storage, a second method for storing said objects which are processed together from volatile storage to nonvolatile storage, and a third method for generating in volatile storage a copy of said objects which are processed together in volatile storage, said unit of work frame further including pointers to said copy of said objects which are processed together; and

means, responsive to a request to process selected ones of said plurality of objects together, for generating an instance of said unit of work object class, said instance of said unit of work object class including a pointer to each selected object, such that said selected

objects are loaded from nonvolatile storage to volatile storage by invoking said first method on said instance of said unit of work object class, and said selected objects are stored in nonvolatile storage after processing by invoking said second method on said instance of said unit of work object class; and

means responsive to completion of a first step of said plurality of steps on said selected objects, for generating a copy of said selected objects in volatile storage, in modified condition from said first step, by invoking said third method, such that said second step is performed on said copy of said selected objects in nonvolatile storage.

41 04498131 Feb 5 1985 395/412 Data processing system having
May 22 1981 addressing mechanisms for
processing object-based
information and a protection
scheme for determining access
rights to such information

Inventor: Bratt; Richard G. et al.
Assignee: Data General Corporation
Abstract:

A digital data processing system has a memory organized into objects containing at least operands and instructions. Each object is identified by a unique and permanent identifier code which identifies the data processing system and the object. The system utilizes unique addressing mechanisms the addresses of which have object fields, offset fields and length fields for specifying the location and the total number of bits of an addressed object. The system uses a protection technique to prevent unauthorized access to objects by users who are identified by a subject number which identifies the user, a process of the system for executing the user's procedure, and the type of operation of the system to be performed by the user's procedure. An access control list for each object includes an access control list entry for each subject having access rights to the object and means for confirming that a particular active subject has access rights to a particular object before permitting access to the object.

Exemplary Claim [1]:

1. In a digital computer system including processor means for performing operations on operands, memory means for storing operands and instructions for directing said operations, said memory means being organized into objects for containing operands and instructions, bus means for conducting operands and instructions between said memory means and said processor means, and I/O means for conducting operands between said digital computer system and devices external to said digital computer system, said processor means comprising:

ALU means connected to said bus means for performing said operations,

addressing means connected to said bus means for providing addresses for controlling the transfer of operands and instructions between said memory means and said processor means, each of said addresses comprising

an object field for identifying an object,

an offset field for specifying a first number of information bits of offset relative to the start of said object, and

a length field for specifying an second number of information bits of said object following said first number of information bits to be transferred between said memory means and said processor means,

and

microcode control means for storing sequences of microinstructions for controlling said processor means and responsive to said instructions for providing said sequences of microinstructions to said processor means,

said ALU means including

general register file means connected to said bus means for storing selected operands and addresses,

said general register file means comprising a plurality of vertically ordered registers vertically divided into three parallel-operating and addressed parts,

a first part of said general register file means comprising first register file means for storing object fields of said addresses,

a second part of said general register file means comprising second register file means for storing offset fields of said addresses and said operands, and

a third part of said general register file means comprising third register file means for storing length field of said addresses, and

address ALU means connected to said general register file means and to said bus means and responsive to first sequences of microinstructions for performing operations on said addresses,

said objects containing operands as data objects and instructions as procedure objects,

and said processor means including

protection means for preventing a user. currently using said digital computer system to execute a program comprising at least one procedure object, from obtaining unauthorized access to objects of another user, said protection means including

subject number memory means responsive to the operation of said processor means for storing a currently active subject number of a plurality of subject numbers,

each subject number corresponding to and identifying one of a plurality of subjects, each subject comprising the combination of (1) a user of said digital computer system, (2) a process of said digital computer system for executing a procedure of said user, and (3) the type of operation to be performed by said digital computer system in response to an instruction of a procedure of said user, and

said currently active subject number identifies (1) the user currently utilizing said digital computer system, (2) the process currently executing a procedure of said user's program, and (3) the type of operation to be performed by said digital computer system in response to a current instruction of a procedure of said user's program,

protection memory means for storing at least one access control list, each access control list corresponding to an object and comprising at least one access control entry, each access control entry corresponding to a subject having access rights to said object and containing the access rights of said subject to said object, and

access right confirmation means responsive to a currently active subject number and to the operations of said processor means for indexing said protection means in response to a current instruction of a procedure of said user's program when said said current instruction requests access to an object and for comparing the access rights of the currently active subject to the access control list associated with said object.

42 05644766 Jul 1 1997 395/620 System and method for managing a
Aug 23 1996 hierarchical storage system
through improved data migration

Inventor: Coy; Henry Robert et al.

Assignee: International Business Machines Corporation

Abstract:

A system and method are provided for preserving spacial and temporal locality of sets of related objects when moving the sets within a storage hierarchy via a common server. The appropriate meta data is gathered to

track the spacial and temporal locality of the sets of objects being moved within the storage hierarchy and the algorithm uses the meta data to preserve the spacial and temporal locality when moving the objects. A collection of logically clustered data objects is identified. The logical cluster is then moved down through the storage hierarchy together to be stored in less costly storage devices. The logical cluster of data objects is then retrievable more efficiently as a whole when requested.

Exemplary Claim [8]:

8. A computerized distributed hierarchical data storage management system comprising:

a server computer system having a central processing unit and memory;

a plurality of client computer systems each having a central processing unit and a non-volatile storage device, in communication with the server;

a hierarchical data storage system in communication with the server and having a plurality of levels of media instances wherein data is stored more economically on a lower level;

means for identifying a logical cluster of data objects in a first level;

means for identifying at least one media instance in a lower level for storing said logical cluster by first attempting to identify at least one media instance already containing other data objects from said logical cluster, second, attempting to identify at least one empty media instance and third, identifying a media instance with the largest available space; and

storage means for storing said logical cluster in a minimum number of identified media instances.

43 05655101 Aug 5 1997 395/475 Accessing remote data objects in a
May 27 1994 distributed memory environment
using parallel address locations
at each local memory to reference
a same data object

Inventor: O'Farrell; William G. et al.

Assignee: International Business Machines Corporation

Abstract:

A mechanism and method for accessing remote data objects in a distributed memory environment is disclosed. In the distributed memory environment, a number of parallel processors are remote from each other and each has memory storage capacity with parallel address locations. For each data object stored in the local memory of a processor, that processor stores a variable at a specific location pointing to the address of the data object in its local memory. At the same (parallel) location in the local memories of all remote parallel processors, a variable pointing to the home processor for the data object is stored. In order to access the data object, it is only necessary to locate the identifier variable in the processor processing a program pointing to the processor having the data object in storage. Because of the parallel address space, the location will automatically point directly to the same address location in the home storage processor, and this address location contains the address pointing to the actual location of the data object in the local storage of that processor. Alternate means for updating the variable information on data object storage is also provided. In one system, changes are broadcast throughout the parallel processor environment while in the other system, changes are merely retained on the home processor or the home processor and one remote parallel processor responsible for recording changes in that region of the distributed memory environment. When accessed by another remote parallel processor, the change information will be

transmitted.

Exemplary Claim [1]:

1. A composite mechanism for accessing remote data objects in a distributed memory environment having a plurality of parallel processors remote from each other, each processor having a local memory with parallel address locations, wherein for each data object having a storage location in the local memory of a first processor, the mechanism comprises:

a first variable storage repository at a first address location in the local memory of the first processor pointing to said object's storage location in the memory of the first processor; and

a second variable storage repository at the first address location in the local memory of at least one remote parallel processor pointing to the first processor, whereby the first address location in the local memory of the first processor and the first address location in the local memory of the at least one remote parallel processor have a same corresponding address location.

44 04575797 Mar 11 1986 395/385 Digital data processing system
May 22 1981 incorporating object-based addressing and capable of executing instructions belonging to several instruction sets

Inventor: Gruner; Ronald H. et al.

Assignee: Data General Corporation

Abstract:

A digital computer system having a memory system organized into objects for storing data and a processor for processing data in response to instructions. An object identifier is associated with each object. The memory system responds to logical addresses for data which specify the object containing the data and the offset of the data in the object. The objects include procedure objects and data objects. The procedure objects contain procedures including the instructions. Each instruction contains an operation code which belongs to one of several sets of operation codes. All instructions in a single procedure belong to a single operation code set, and associated with each procedure is an operation code set identifier specifying the operation code set to which the instructions in the procedure belongs. Operation codes are decoded in the processor in response to both the operation code and a dialect value representing the operation code set identifier for the operation code set to which the operation code belongs. The dialect value changes only on execution of a call operation or a return operation.

Exemplary Claim [1]:

1. A digital computer system comprising:

(1) memory means for storing and providing items of data, said items of data including instructions, each one of said instructions containing an operation code of a plurality of operation codes, said operation codes belonging to a plurality of functionally different operation code sets, said operation codes in a given one of said operation code sets being definable solely with reference to said given operation code set, and said instructions having common formats; and

(2) processor means connected to said memory means for receiving said items including said instructions and responding to each said received instruction by performing said operation defined for said operation code in said received instruction in said operation code set to which said operation code in said received instruction belongs.

45 05263165 Nov 16 1993 395/490 System for providing user access
Feb 15 1990 control within a distributed data

processing system having multiple resource managers

Inventor: Janis; Frederick L.

Assignee: International Business Machines Corporation

Abstract:

The method of the present invention may be utilized to provide user access control for a plurality of resource objects within a distributed data processing system having a plurality of resource managers. A reference monitor service is established and a plurality of access control profiles are stored therein. Thereafter, selected access control profile information may be communicated between the reference monitor service and a resource manager in response to an attempted access of a particular resource object controlled by that resource manager. A resource manager may utilize this communication technique to retrieve, modify, or delete a selected access control profile, as desired. Further, the resource manager may utilize this communication technique to control access to a resource object by utilizing the information contained within the access control profile to determine if the requester is authorized to access the resource object and whether or not the requester has been granted sufficient authority to take selected actions with respect to that resource object. In a preferred embodiment of the present invention, each access control profile may include access control information relating to a selected user; a selected resource object; a selected group of users; a specified level of authority associated with a selected user; a selected set of resource objects; or, a predetermined set of resource objects and a selected list of users each authorized to access at least a portion of said predetermined set of resource objects.

Exemplary Claim [1]:

1. A computer implemented method of providing variable authority level user access control for a plurality of resource objects within a distributed data processing system having at least one reference monitor service and a plurality of resource managers associated with said plurality of resource objects, each of said plurality of resource managers controlling access to different selected ones of said resource objects, each of said resource managers associated with a reference monitor service, said method comprising the computer implemented steps of:

storing a plurality of unique access control profiles within each said reference monitor service, wherein selected ones of said plurality of access control profiles each include an identification of a selected user and a specified level of authority associated with said selected user;

querying an associated reference monitor service by a selected one of said resource managers in order to vary access control for a particular resource object by a selected user, wherein access to said particular resource object is controlled by said selected resource manager;

transmitting a selected access control profile associated with said selected user from said associated reference monitor service to said selected one of said resource managers if said selected access control profile existed in said associated reference monitor service; if not, attempting to retrieve said selected access control profile from another said reference monitor service and thereafter transmitting said retrieved access control profile to said selected one of said resource managers;

utilizing said selected resource manager to selectively modify said access control information in said selected access control profile; and

storing said selectively modified access control information in said selected access control profile within an associated reference monitor service wherein subsequent access to said particular resource object may be variably controlled.

46 05263158 Nov 16 1993 395/601 Method and system for variable
Feb 15 1990 authority level user access
control in a distributed data
processing system having multiple
resource manager

Inventor: Janis; Frederick L.
Assignee: International Business Machines Corporation
Abstract:

Variable authority level user access control for a plurality of resource objects within a distributed data processing system having a plurality of resource managers. A reference monitor service is established and a plurality of access control profiles are stored therein, each including an identification of a selected user and a specified level of authority associated with that selected user. Thereafter, selected access control profiles are exchanged between the reference monitor service and a resource manager in response to an attempted access of a particular resource object controlled by that resource manager. The resource manager may then control access to the resource object by utilizing the exchanged access control profile to determine the extent access is permitted by means of the specified level of authority contained therein. In a preferred embodiment of the present invention, the access intent of a selected user is determined in conjunction with an attempted access of a particular resource object and stored. Thereafter, a comparison of the stated access intent with the specified level of authority contained within the access control profile may be utilized to grant or deny access.

Exemplary Claim [1]:

1. A computer implemented method of providing variable authority level user access control for a plurality of resource objects within a distributed data processing system having at least one reference monitor service and a plurality of resource managers associated with said plurality of resource objects, each of said plurality of resource managers controlling access to different selected ones of said resource objects, each of said resource managers associated with a reference monitor service, said method comprising the computer implemented steps of:

storing a plurality of unique access control profiles within each said reference monitor service, wherein selected ones of said plurality of access control profiles each include an identification of a selected user and a specified level of authority associated with said selected user;

querying an associated reference monitor service by a selected one of said resource managers in response to an attempted access of a particular resource object by a selected user, wherein access to said particular resource object is controlled by said selected resource manager;

transmitting a selected access control profile associated with said selected user from said associated reference monitor service to said selected one of said resource managers if said selected access control profile existed in said associated reference monitor service; if not, attempting to retrieve said selected access control profile from another said reference monitor service and thereafter transmitting said retrieved access control profile to said selected one of said resource managers;

utilizing said selected resource manager to control access to said particular resource object in accordance with access control information in said selected access control profile; and

denying access to said particular resource object by said selected user in response to a failure to retrieve said selected access control profile.

47 05450581 Sep 12 1995 395/609 System for copying from one
Oct 11 1994 database management system to

another by translating authorization statements

Inventor: Bergen; Dianne E. et al.
Assignee: International Business Machines Corporation
Abstract:

A computer-implemented method is provided which helps to render the differences between relational database management system (RDBMS) types transparent. The computerized process enables database objects and authorizations on objects, as well as user privileges, to be moved or copied from one RDBMS to another within a distributed relational database environment without requiring a database administrator to have expertise in every RDBMS type in that environment. The computerized process is achieved using three inter-related and inter-dependent tables, a translation table, an implications table and a composition table which are stored in a computer memory and accessed through a computer. These tables provide a mapping matrix between database objects of a plurality of RDBMSs.

Exemplary Claim [1]:

1. A system for copying a database object from a source database to a target database, wherein said target database may differ from said source database in an authorization scheme for database objects and in authorization statements used to implement the authorization scheme, said system comprising:

a translation table for mapping a plurality of source authorizations of said source database to a plurality of target authorizations of said target database;

means for identifying, from said translation table, a corresponding target authorization mapped to one of said source authorizations associated with a source database object to be copied to said target database;

a composition table for automatically composing at least one authorization statement for implementing said corresponding target authorization in a syntax of said target database; and

means for copying said source database object from said source database to said target database.

48 05414852 May 9 1995 395/674 Method for protecting data in a
Oct 30 1992 computer system

Inventor: Kramer; Paul H. et al.
Assignee: International Business Machines Corporation
Abstract:

A data processing system include a plurality of data objects which are accessible by application programs through a system level interface. Each data object has an associated user access list. In addition, each object has at least one key indicating which applications can access that object. The key is preferably maintained in a protected storage area, accessible only by the low level system interface. Both the application identifier key and the user who invoked that application must match the identifier information in the data object for access to be allowed to that object. If an unauthorized user attempts access to the data object through the correct application, or an authorized user attempts access through an incorrect application, access to the data object will be denied by the low level interface.

Exemplary Claim [1]:

1. In a data processing system, a computer-implemented method for controlling access to a plurality of data objects, comprising the steps of:

when each of the plurality of objects is created, associating with

such object at least one user identifier, and at least one list of data manager identifiers;

invoking a first data manager using a user identifier to identify an invoking user;

invoking a second data manager from the first data manager; and

when the second data manager later attempts to access a data object, allowing such access only if one of the lists of data manager identifiers associated with the data object contains both the first and second data managers, and the first data manager has been invoked with a user identifier associated with the object.

49 05241682 Aug 31 1993 395/200.79 Border node having routing and functional capability in a first network and only local address capability in a second network
Apr 18 1991

Inventor: Bryant; David B. et al.
Assignee: International Business Machines Corporation
Abstract:

A method and apparatus for interconnecting multiple data processing networks, each data processing network including: multiple network nodes having routing and functional capability within a data processing network; and, multiple endpoint nodes, each including only local address capability. Each network node may be connected to multiple endpoint nodes and other network nodes; however, connection is only permitted to network nodes within the same data processing network. A border node is established for interconnection between two data processing networks. The border node includes a network node interface having routing and functional capability within a first data processing network and an endpoint node interface having local address capability. When interconnected between the first data processing network and a network node within a second data processing network, the border node emulates a network node within the first data processing network while appearing as an endpoint node within the second data processing network, permitting full connectivity between the two networks. The border node then maintains routing information for communication between the two data processing networks in two segments. A first segment details the route between the first data processing network and the border node while a second segment details the route between the border node and a node within the second data processing network.

Exemplary Claim [1]:

1. A method for interconnecting a plurality of data processing networks, each said data processing network including: a plurality of network nodes, each said network node having routing and functional capability within said data processing network, and, a plurality of endpoint nodes, each said endpoint node having only local address capability, each said network node being connected to another said network node only within the same data processing network and to a plurality of said endpoint nodes within any said data processing network, said method comprising the steps of:

establishing a border node within a first one of said data processing networks;

establishing within said border node a network node interface having routing and functional capability for said first data processing network;

establishing within said border node an endpoint node interface having local address capability;

linking sad endpoint node interface of said border node to a selected network node within a second one of said data processing networks, identifying said border node as an endpoint node having local address

capability within said second data processing network;

transmitting directory search requests from within said first data processing network to said selected node within said second data processing network; and

adding local address information to each said directory search request at said border node prior to transmitting each said directory search request to said selected network node within said second data processing network, said local address information specifying at least said link between said border node and said selected network node within said second data processing network.

50 05564113 Oct 8 1996 395/604 Computer program product for Jun 7 1995 rendering relational database management system differences transparent

Inventor: Bergen; Dianne E. et al.

Assignee: International Business Machines Corporation

Abstract:

A computer-implemented method is provided which helps to render the differences between relational database management system (RDBMS) types transparent. The computerized process enables database objects and authorizations on objects, as well as user privileges, to be moved or copied from one RDBMS to another within a distributed relational database environment without requiring a database administrator to have expertise in every RDBMS type in that environment. The computerized process is achieved using three inter-related and inter-dependent tables, a translation table, an implications table and a composition table which are stored in a computer memory and accessed through a computer. These tables provide a mapping matrix between database objects of a plurality of RDBMSs.

Exemplary Claim [1]:

1. A computer program product for use with one or more database management systems, said computer program product comprising:

a computer usable medium having computer readable program code means embodied in said medium for establishing at least one target authorization to a target database derived from at least one source authorization to a source database the source database being in communication with the target database and a workstation, the source database, target database, and workstation being in a computer system having a processor, memory, and data storage device, said source database using a source database management system that is different from a target database management system used by the target database, said computer readable program code means comprising:

computer readable program code means for causing a computer to map said source authorization to a corresponding target authorization of said target database management system;

computer readable program code means for causing a computer to identify at least one implication for said source authorization having a corresponding target authorization having non-identical function to said source authorization; and

computer readable program code means for causing a computer to automatically compose at least one authorization statement for implementing said corresponding target authorization in a syntax of said target database management system.

Confidentiality:

EDS SPO considers this communication confidential. None of the information contained in this report is shared with any third parties.

Important Notice:

Please review this report and if the product is not satisfactory, notify EDS SPO within 24 hours. Please include the information in the Sales Order Summary as part of your message. This information is vital in identifying reports run by customers.

&